Andreas Dabnell

DESIGNING AND PROGRAMMING AUTOMATION IN DISTRIBUTED CONTROL SYSTEM ENVIRONMENT

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Degree programme of Information Technology
May 2013

Centria
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Unit	Date	Author
Kokkola-Pietarsaari	May 2013	Andreas Dabnell
Degree programme Information Technology		
Title DESIGNING AND PROGRAMMING AUTOMATION IN DISTRIBUTED CONTROL SYSTEM ENVIRONMENT		
Instructor M.Eng. Hannu Ala-Pöntiö		**Pages** 56 + 11
Supervisor M.Sc. (Tech.), Matias Paloranta		

This thesis studies distributed control systems on a general level followed by experimental parts of designing and programming assigned cases. The objective of those cases were to create control applications within the system for a hydrogen sulfide desuplhurization system and a sulfur dioxide distribution tank.

The thesis was commissioned by Freeport Cobalt Ltd. chemical plant as part of plant-wide automation system upgrade. The goal was to create applications which would be efficient, safe, and user-friendly.

The experimental part was completed with the ABB Industrial IT 800xA distributed control system and its composition was also used as a reference throughout the theoretical part of this thesis. The control applications were successfully created and introduced to everyday use at the plant.

Key words
ABB 800xA, distributed control system, programming, control theory, instrument parametrization, control application

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Kokkola-Pietarsaari	**Aika** Toukokuu 2013	**Tekijä** Andreas Dabnell
Koulutusohjelma Tietotekniikan koulutusohjelma		
Työn nimi AUTOMAATION SUUNNITTELUA JA OHJELMOINTIA HAJAUTETUSSA OHJAUSJÄRJESTELMÄYMPÄRISTÖSSÄ		
Työn ohjaaja Ins. (YAMK) Hannu Ala-Pöntiö		**Sivumäärä** 56 + 11
Työelämäohjaaja D.Ins. Matias Paloranta		

Tämä opinnäytetyö tutkii hajautettuja ohjausjärjestelmiä yleisellä tasolla, jota seuraa kokeelliset osuudet automaation suunnittelusta ja ohjelmoimisesta määritellyissä tapauksissa. Tapaukset olivat luoda järjestelmän sisällä ohjaussovellukset rikkivedyn rikinpoistojärjestelmälle, sekä rikkidioksidin jakelusäiliölle.

Tämän opinnäytetyön tilasi Freeport Cobalt Oy kemikaalilaitos osana laitoksen automaatiojärjestelmän uusintaa. Tavoitteena oli luoda sovellukset jotka olisivat suorituskykyisiä, turvallisia, ja käyttäjäläheisiä.

Kokeellinen osuus suoritettiin ABB Industrial IT 800xA hajautetulla ohjausjärjestelmällä sekä sen rakennetta käytettiin myös viitekehyksenä teoriaosuudessa. Sovellukset saatiin onnistuneesti luotua ja ne otettiin jokapäiväiseen käyttöön tehtaalla.

Asiasanat
ABB 800xA, hajautettu ohjausjärjestelmä, ohjelmointi, säätöteoria, instrumentoinnin parametrointi, ohjaussovellus

ABBREVIATIONS

DCS	Distributed Control System, automation system commonly used in process automation
I/O	Input / Output, describes external signal connections
HMI	Human Machine Interface, a terminal used to interact with devices
CI	Communication Interface, used to connect external devices to target
TCP/IP	Transmission Control Protocol / Internet Protocol, communication protocol used in the internet and other networks
FF	Foundation Fieldbus, a fieldbus standard
DNS	Domain Name System, naming system for resources connected to the internet or other private network
OPC	Object Linking and Embedding for Process Control, communication standard for automation systems and devices
CPU	Central Processing Unit, hardware within a device that carries out instructions of a computer program
Profibus DP	Process Field Bus Decentralized Peripherals, used to operate sensors and actuators in automation systems
PID controller	Proportional-Integral-Derivative controller, used to automatically direct a quantity to a desired setpoint

A/D converter	Analog-to-Digital converter, converts a continuous physical quantity to a digital number
RNRP	Redundant Network Routing Protocol, used to switch to a secondary network in a duplicated system
MCC	Motor Control Center, an assembly of several motor control units which can contain a set of auxiliary devices
FBD	Function Block Diagram, a graphical programming language used commonly in automation, IEC 61131-3 standard
ST	Structured Text, a programming language part of IEC 61131-3 standard
SFC	Sequential Function Chart, a graphical programming language used to describe sequential behavior (IEC 61131-3)
IEC 61131-3	International standard for programmable logic controllers, includes LD, FBD, ST, IL and SFC
ATEX	ATmosphères EXplosives, an EU directive describing allowed work and equipment in an explosive atmosphere
RTDB	Real Time DataBase, a relational database designed for process information management and history recording

ABSTRACT
TIIVISTELMÄ
ABBREVIATIONS
TABLE OF CONTENTS

1 INTRODUCTION 1

2 FREEPORT COBALT LTD 3

3 DISTRIBUTED CONTROL SYSTEM 5
3.1 System architecture 6
3.1.1 System communications 7
3.1.2 Servers 8
3.1.3 Controller functions 10
3.1.4 I/O signals and module types 11
3.1.5 Workstations and process graphics 12
3.1.6 Network and device redundancy 13
3.2 Operations software 14
3.2.1 ABB System 800xA 14
3.2.2 Operator Workplace 14
3.2.3 Engineering Workplace 16
3.2.4 Control Builder M Professional 17
3.2.5 Graphics Builder 18
3.3 Programming languages 19
3.3.1 Function Block Diagram 20
3.3.2 Structured Text 21
3.3.3 Sequential Function Chart 22
3.4 Summary 23

4 AUTOMATION OF H_2S DESULPHURIZATION SYSTEM 24
4.1 Operation principle 25
4.1.1 Instrumentation 26
4.1.2 Control loops and control theory 27
4.2 Practical programming phases 31
4.2.1 Process graphics display 33
4.2.2 Interlocks 35
4.2.3 Cascade control connections 36
4.2.4 System sequence start and stop 37
4.3 Summary 39

5 AUTOMATION OF SULFUR DIOXIDE DISTRIBUTION TANK 40
5.1 Operation principle 40
5.1.1 Instrumentation 41
5.1.2 Control loops 42
5.2 Practical programming phases 43

5.2.1 Process graphics display **43**
5.2.2 Interlocks **44**
5.2.2 Calculated consumption measurement **44**
5.2.3 Automatic control loop setpoint drop **48**
5.3 Summary **50**

6 CONCLUSIONS **51**

REFERENCES **54**

GRAPHS

GRAPH 1. Process chart of Freeport Cobalt Ltd. 4
GRAPH 2. Example DCS system architecture 6
GRAPH 3. Comparison of instrument connection setups 8
GRAPH 4. Example of remote client access 9
GRAPH 5. Simple controller rack setup 11
GRAPH 6. Process graphics display 12
GRAPH 7. Physical view of a full redundant network 13
GRAPH 8. PI Control loop faceplate 15
GRAPH 9. Engineering Workplace screen 17
GRAPH 10. Liuotto -project in Control Builder 18
GRAPH 11. Example of a CTUD counter block 21
GRAPH 12. A sequence structure 23
GRAPH 13. A bioscrubber system for removal of H_2S 24
GRAPH 14. Indoor view of the desulphurization system 26
GRAPH 15. P6109 closed control loop 27
GRAPH 16. Q6107 and F6111 cascade control 29
GRAPH 17. Example pole-zero plot 30
GRAPH 18. Steps for building a project in ABB 800xA 31
GRAPH 19. Control module parameters of PID controller Q6107 32
GRAPH 20. A controller rack with several CI modules connected 33
GRAPH 21. An I/O cluster with redundant fieldbus slave units 33
GRAPH 22. Configuration of H_2S desuplhurization system graphics 34
GRAPH 23. TOf Function Block timing diagram 35
GRAPH 24. Group sequence status display 38
GRAPH 25. Simplified operational description of the SO_2 distribution tank 40
GRAPH 26. SO_2 distribution tank control loops 42
GRAPH 27. Princible of the code used to calculate consumption of SO_2 45
GRAPH 28. ShiftRegisterReal Function Block 46
GRAPH 29. Timing diagram of PulseGenerator Function Block 46
GRAPH 30. Flowchart describing the events in the automatic setpoint drop 48

TABLES

TABLE 1. Examples of basic language functions 19
TABLE 2. Instrument list of the H_2S desulphurization system 26

TABLE 3. Instrument list of the SO_2 distribution tank 42
TABLE 4. Function Blocks used in the consumption measurement code 47

1 INTRODUCTION

Process automation is the control of blending and treatment of raw materials that result in a continuous stream of product. This is achieved by, depending on the plant size, hundreds to tens of thousands of sensors constantly measuring and monitoring process quantities. These sensors are connected to the distributed control system which processes the information in a way defined by program code. Once processed, the system controls actuators which change the process flow. The most essential purpose of process automation is to maximize production efficiency at as little cost as possible. Due to the increasing complexity of manufacturing methods, control of rapidly changing conditions, for example, some tasks would not be possible without high level modern automation. Because of these variables, the level of automation in process industry is constantly on the rise and engineers involved must specialize in increasingly narrower areas of expertise.

Also in the case of this thesis, the automation system consists of a large amount of technologies from different areas of engineering and sciences. For example, information technology is applied in communications, and servers. and electrical engineering is applied in variable frequency drives, and motors. This thesis does not focus solely on a single area, but studies the composition of distributed control systems on a general level, and the practical configuration of control applications. In order to successfully design and program automation applications, an extensive engineering knowledge background of the system has to be achieved.

This thesis was commissioned by Freeport Cobalt Ltd, a chemical plant located in Kokkola, Finland. The plant has currently two different automation systems in use; the older ABB Advant and the newer ABB 800xA. The lifecycle of ABB's Advant system is nearing its end and is being replaced by the 800xA. The update process has been ongoing for a while, and is completed in sections at a time. This is because updating the whole system is nearly impossible without long plant downtime periods. The purpose of the experimental part of this thesis was to design and reprogram the automation of two entireties which were a hydrogen sulfide desulphurization system and a sulfur dioxide distribution tank. The goal was to successfully build control applications for both and to create simple control graphics for the use of the process operators.

Throughout the practical work, process operators were considered as the end users of the applications and their needs and efficiency was considered a priority. Even though the applications could be complex, the visibility of these applications to the operators should be clearly structured and simple to use. Those factors would be facilitated by clear documentation and easily interpreted process graphics.

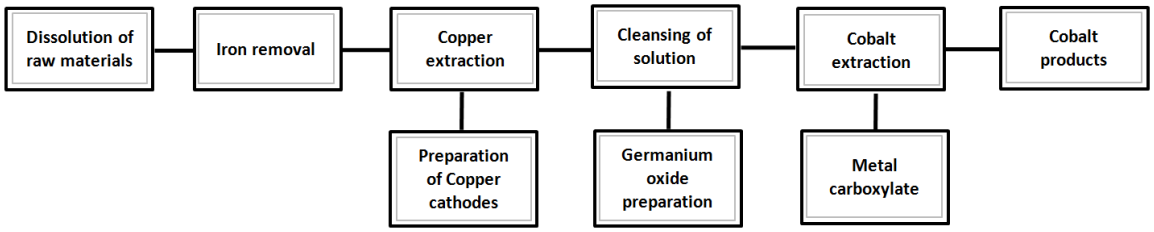
This work is divided into three main chapters. First, general theory of distributed control systems and related devices, methods and programming were studied. The theory part should give a good introduction to the skill and knowledge that was required to complete the practical work of this thesis. The practical work of designing and programming automation in the system is discussed in the two following chapters, 4 and 5.

2 FREEPORT COBALT LTD

Freeport-McMoRan Copper & Gold Inc.(NYSE: FCX) is an international mining company with headquarters in Phoenix, Arizona. FCX operates geographically diverse assets with proven and potential reserves of copper, gold and molybdenum. FCX is the world's largest publicly traded copper producer and has a dynamic portfolio of operating, expansion and growth projects in the copper industry. FCX is also the world's largest producer of molybdenum and a significant gold producer. The company's global workforce includes approximately 34,000 employees. (Freeport-McMoran Copper & Gold Inc. 2013.)

Freeport Cobalt Ltd. (previously OMG Kokkola Chemical Ltd.) is a chemical plant located in Kokkola, Finland and is a new acquisition by FCX. With around 400 employees, the plant produces inorganic salts, oxides and various types of cobalt powders from cobalt and nickel, additionally metallic carboxylates and copper cathode plates. As raw materials, the plant uses enriched cobalt, cobalt-containing intermediate products, and metal granule that is imported from Africa and is refined on site. The final products are used in a variety of applications, such as in the battery industry, petrochemicals industry, in tungsten carbide blades, and many other. (OMG Kokkola Chemicals Ltd. 2011.)

The plant process flow is presented in Graph 1. The process starts with dissolution, where cobalt is separated from other metals contained in the raw material. Copper contained in the raw material is separated from the cobalt by electrolysis, which will result in pure copper cathodes. Other valuable metals, such as germanium and nickel, are also separated and sold. The cobalt is cleansed in the solvent extraction process, where impurities are removed before the end product preparation. End products, cobalt chemicals and powders are manufactured from the pure cobalt solution. Cobalt powder is manufactured by reducing the cobalt into metal powder with the use of hydrogen gas. Metal carboxylate products are also included in the end products. (OMG Kokkola Chemicals Ltd. 2011.)



GRAPH 1. Process chart of Freeport Cobalt Ltd. (OMG Kokkola Chemicals Ltd. 2011.)

The development of products and manufacturing methods, and a fully automated, continuous process is an essential part of maintaining company competitiveness in the global market. Freeport Cobalt invests a great deal in these aspects and aims to develop products to match customer needs. Freeport Cobalt Ltd produces analysis and testing services for the needs of production, quality control, research and development. Advanced modern methods are used in the laboratory for the examination and metal analysis for studying the physical properties of the products. (OMG Kokkola Chemicals Ltd. 2011.)

Freeport Cobalt Ltd. Supplies products worldwide to over 50 countries. Most of the logistics and transport is done by trucks, but also shipping and rail transports are utilized for American and Asian markets. The plant has a central warehouse from where most of the products are delivered to customers in Europe. The other main market areas have their respective intermediate warehouses. (OMG Kokkola Chemicals Ltd. 2011.)

3 DISTRIBUTED CONTROL SYSTEM

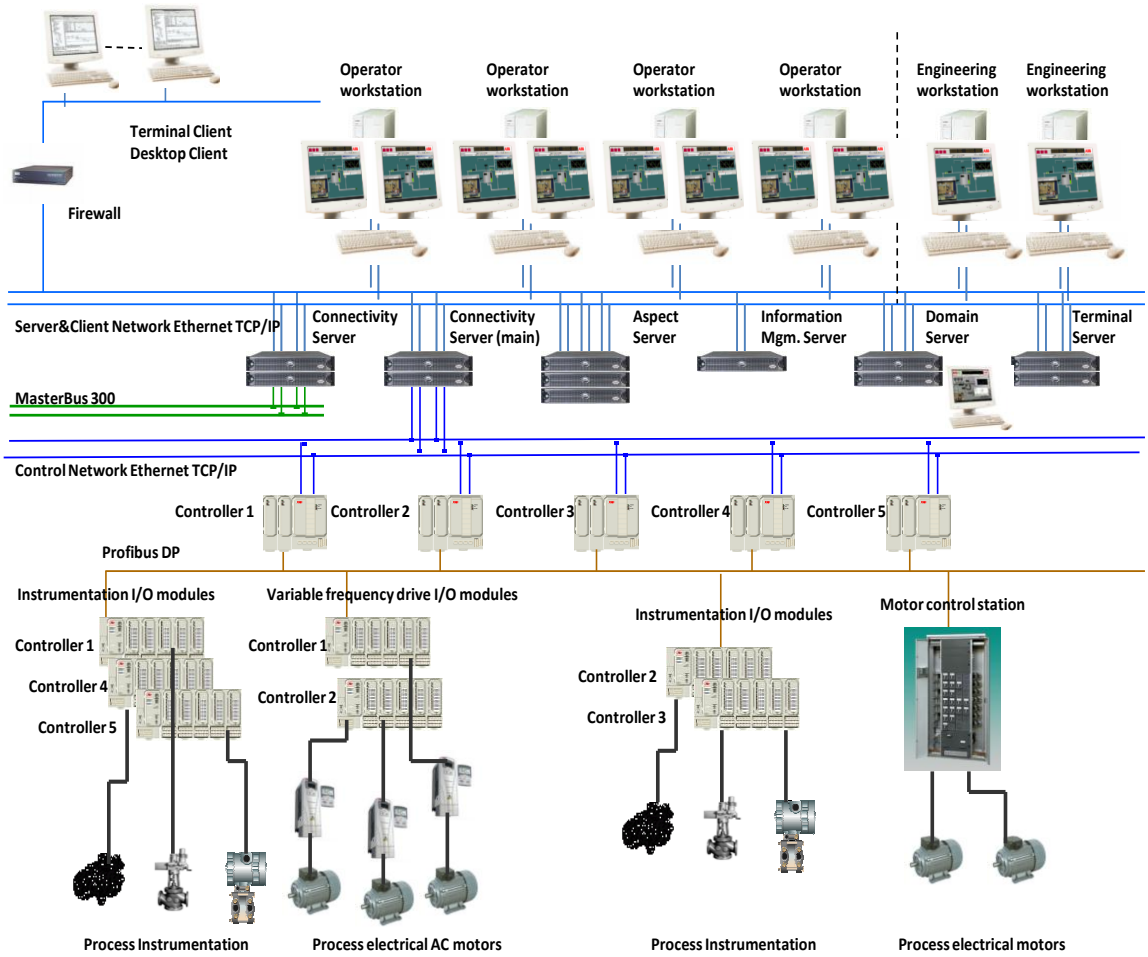
Distributed Control System, or DCS, is a broad term used by many decentralized electrical systems. In this case DCS is referred to the industrial process automation system. The applied automation system at Freeport Cobalt Ltd. utilizes distributed control architecture. A distributed control system consists of several processor units called controllers geographically spread throughout the plant. Each controller has its own sub-system that is independent of the other controllers. The DCS architecture is characterized by these conventional I/O (input/output) subsystems, in which racks of I/O modules are networked to their respective centralized controller via an I/O subsystem network. Controllers are connected to system connectivity servers which are read by operator computer terminals for information and interaction of the plant processes. (Berge, J. 2002.)

Controllers are usually divided by plant sections, for example, one for refining process and another for chemicals production. The physical process instruments and motors are connected to I/O modules by either current loops of 4-20mA or digital pulses. The I/O modules are arranged in clusters which are then connected by a fieldbus to a controller.
Handling of the information received from field instruments is defined by the program code that is downloaded into the controller memory. This way connection interrupts between the server and controller will not stop the process flow entirely. (Berge, J. 2002.)

The operator and engineering workstations are in the uppermost level of the system hierarchy. The operator workstations are normally situated in the plant control rooms. From these terminals the operators can monitor and control the information received from the process instruments. The information is incorporated into process graphics which closely correspond to the actual field environment. This makes an easy to use, efficient HMI (Human-Machine-Interface). Engineers configure the system from the engineering workplaces with the help of supplier specific programming tools and software. (Festo Didactic GmbH & Co. 1997.)

3.1 System architecture

The system architecture specifies the relations of the different parts of the automation system. In Graph 2, the composition of the system is portrayed in a tree-like structure. In the roots and top are the terminal devices which do not share further connections than the one. In this case they are the field instruments, actuators, motors and computer terminals. In traditional I/O wiring, each device has its own physical wiring connection to the system I/O ports. I/O modules are arranged in a cluster and the information is collected in a 240 byte package in the fieldbus slave unit. The information packages are collected to the fieldbus communication interfaces which are in a series with a controller. The controller finally reads the CI units in fixed intervals. Data from the controllers is read via the connectivity server, from which the operator terminals can monitor and control relevant plant data. (Paloranta 2012.)

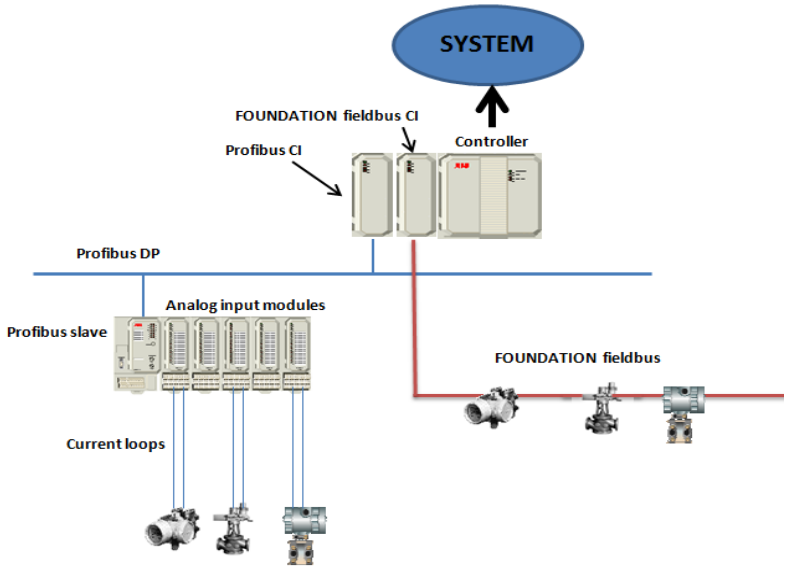


GRAPH 2. Example DCS architecture (adapted from Paloranta, M 2012.)

3.1.1 System communication

Process automation systems utilize several different communication protocols and the topology varies by supplier and plant. A DCS can often have, in all, as many as four different tiers of networking, each with different technology: device, I/O subsystem, controllers, and plant-wide integration to to business applications. The modern configuration, however, is to have an Ethernet TCP/IP network for the communication between the computer terminals and the servers. TCP/IP is also used to link the controllers to the connectivity server. In decentralized I/O, a fieldbus normally profibus DP, is used for communication between the master profibus Communication Interface in connection with the controller, and the remote slave I/O unit. (PI organization 2013.)

Traditionally, a common method of sending information from an instrument to the system is done by analog 4-20mA current loops, 4mA representing the lowest measurement value and 20mA the highest. This requires a minimum of two wires per instrument which in large plants result in even kilometers of current loop wiring. Information received by the 4-20mA method is limited to measurement value only. Foundation fieldbus, for example, is a technology addressing these issues. FF connects the instruments in a single all-digital serial bus. The increased electronic capabilities of the instruments allow more information to be added to the data package sent in the bus, for example electricity consumption data and maintenance notifications. A disadvantage in the FF system and other fieldbuses is the more vulnerable structure: if the bus is severed in one spot, all the instruments and actuators connected stop working. This raises the need for redundancy of the fieldbus communication. (Fieldbus Foundation 2006.)



GRAPH 3. Comparison of instrument connection setups.

3.1.2 Servers

The servers in the automation system are central information, data and communication hubs. As shown in Graph 2, several different types of servers are used. Each has it's own purpose and function to streamline and clarify operation. This paragraph explains the functions of different servers. Please keep in mind that the setup and naming policies vary by system supplier and this text is to give a general perception of server functions in an automation system.

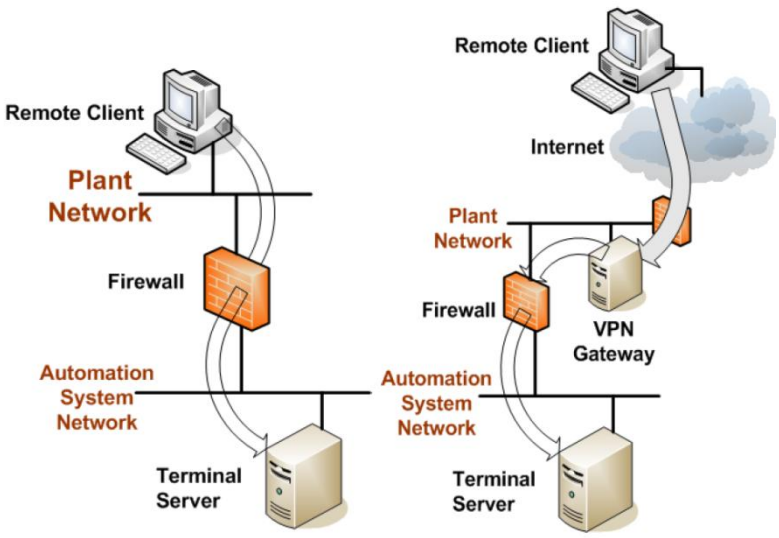
DNS (Domain Name System) is a hierarchical name service for domains and IP addresses. The DNS service enables client nodes on a network to register and resolve DNS domain names and to find domain controller services. DNS can be used for finding the IP address of a node which is only known by name (address resolution) and for finding the name of a node which is known by IP address (name resolution). DNS can also be used for finding domain controller services i.e. which node runs a certain service for the domain.
(ABB Inc. 2012a.)

Aspects servers run the intelligence of the system. Aspects are informational items associated with objects, such as I/O definitions, engineering drawings, process graphics, reports, and trends that are assigned to each object in the system. These servers are the

most important ones at Freeport Cobalt: if they fail to operate, so does the entire plant. Because of this two redundant pairs are applied and separated geographically in order to ensure constant operation. ([6] ABB Inc. 2003.)

Connectivity servers provide access to controllers and other data sources. Several groups of connectivity servers may exist in a system, each serving one set of data sources. Examples of services that run on a connectivity server are OPC related services, and connections to database services. In Graph 2, the secondary connectivity server connects another DCS via a fieldbus. This enables control of two separate systems from the same operator terminals. At Freeport Cobalt, there are several connectivity servers applied, each collecting data from different sources. For example, one is used exclusively for Foundation Fieldbus connectivity and data. (ABB Inc. 2012a.)

Terminal servers handle remote desktop connections and user authentication. Users can have different levels of access permissions; the terminal server restricts clients according to the level of access they have. In Graph 4, remote access is shown from inside the plant network and from the internet. For example, the screen captures in this thesis were captured by connecting to a terminal server from an office network computer by Windows Remote Desktop. Once connected, a new Remote Desktop connection was formed from the terminal server to a automation system engineering client from whereof the screen captures are taken. (ABB Inc. 2012a.)



GRAPH 4. Examples of remote client access (ABB Inc. 2012a.)

Information Management servers log historical process data. This data is trends from the process, electrical history, alarm and event history, and production data. This information can also be transferred into a plant-wide database system. Freeport Cobalt does not utlize separate Information Management servers; history logs are gathered in a separate RTDB system via the connectivity servers. (ABB Inc. 2012a.)

OPC (Open Process Control) assures interoperability between devices from different manufacturers. OPC server converts data from different sources into a standard-based OPC format. This can be utilized in connecting and controlling an automation entirety from a different supplier than the main system. This standard is managed by the OPC foundation. (OPC Foundation 2013.)

3.1.3 Controller functions

Process Automation Controllers are essential to any process automation system. Controllers consist usually of a CPU, PSU, and I/O- and communication modules. Controllers are very dynamic and modules can be added and removed easily. Controller CPU's are digital computers which compute I/O data and run programs designed to control plant functions. PID control functions, for example, are calculated in the CPU. The clock frequency is usually between 24MHz to 500MHz. Because the execution is not continuous, the downloaded program is run through in fixed intervals, for example, 500-1000ms. Communication modules are fieldbus master CI's which connect remote slave devices to the CPU. Support for different fieldbus standards varies by manufacturer. In Graph 5. is an example controller rack with, from the right, a PSU, an empty I/O connection plate, 16-bit digital input module, CPU with Ethernet CI's and a Profibus DP CI. (Paloranta 2013.)

GRAPH 5. Simple controller rack setup

3.1.4 I/O signals and module types

Input/Output modules have the task to send and receive signals from the physical process devices. There are four basic types of I/O: Digital Input/Output modules, and Analog Input/Output modules. As shown in Graph 3, there are alternative methods to receive information from the field devices. At Freeport Cobalt, mostly traditional I/O signals are used and their number is around 30,000.

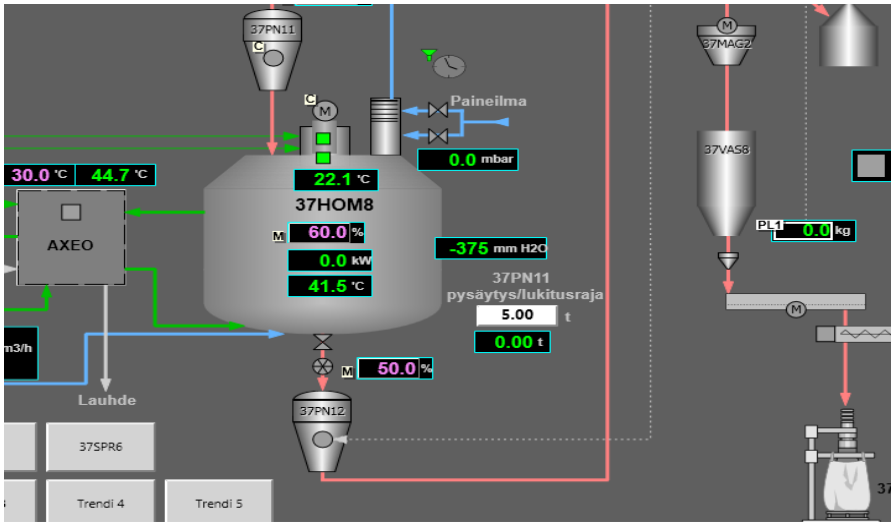
Digital signals are 5VDC electric pulses of varying duration. The signal is interpreted as a logic level signal with two states, 1 and 0. There are no intermediate values and therefore tolerance for rising and falling edge is used. Level “0” might represent a voltage value of 0-2V and “1” a voltage of 3-5V. In some cases the signal is inverted in the program code i.e. 5V is interpreted as a logic value of “0”. An example of a negated Boolean input can be seen in paragraph 3.3.1. This is usually used for security purposes as there is no way of knowing if the signal wire is cut, if the signal is normally at 0V. In process automation, digital signals are mostly used for limit switches which trigger system alarms and events. (University of Alberta 2003.)

Analog signals are current loops with a current value between 4-20mA. The signal constantly circulates between the I/O module and the field device. Usually the signal is used to measure process quantities such as fluid level. The actual level is scaled to 4-20mA, 4mA representing zero value, and 20mA max level. The 4mA “live zero” helps to detect failures in the loop. When the current drops to 0mA it is an indication of either

instrument or wiring failure. The signal goes through an A/D converter in the analog I/O module for interpretation by the program code. In a 16bit converter, for example, the signal is converted to a decimal value of 0-65536, or -32768 - +32768.
(National Instruments Corp. 2012.)

3.1.5 Workstations and process graphics

Operator and engineering workplaces are PC-based computers which have the purpose to work as HMI's. Supervision and operation of the site is done from these terminals. The information from the plant is presented in graphical, dynamic user interfaces which are made to easy to understand and operate. Topologically, the computers are connected to the server/client Ethernet network as clients in a ring configuration. An example of process graphics can be seen in Graph 6. By clicking on an object on the screen, the operator can inter alia change automatic control setpoints. Operators usually only have access to their own respective site graphics and experts are assigned to each process section.
(ABB Inc. 2012c.)

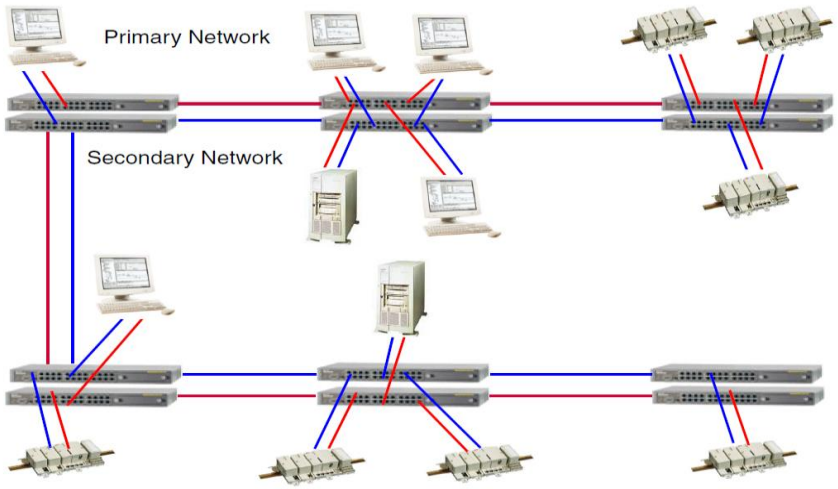


GRAPH 6. Process graphics display

Engineers can configure the system either from special engineering workstations, or any client station by using different user account. Creating process graphics and writing program code is done by special manufacturer supplied software which are examined further in section 3.2. (ABB Inc. 2012c.)

3.1.6 Network and device redundancy

In a continuous process, such as Freeport Cobalt's manufacturing process, even short unscheduled plant downtimes can be very costly. The different process sections are linked together and shutdown of one interrupts material flow to the other. This can effectively stop entire plant operations. The goal is to keep the uptime at a minimum of 98,07% i.e. one week of scheduled downtime per year. Redundancy helps to achieve this goal. Redundancy is the practice of duplicating critical system functions in order to increase the reliability of the system. As seen in Graph 7, all the network components (cables, switches, routers and network adapters) are duplicated. (Paloranta 2013.)



GRAPH 7. Physical view of a full redundant network (ABB Inc. 2012a.)

As long as the primary network paths are working, all process data is sent on that network. The secondary network normally carries no user traffic. This guarantees that network performance is not affected after a network fail over. Both supervision of network paths and fail-over between Primary and Secondary networks are performed by RNRP (Redundant Network Routing Protocol). RNRP is an IPv4 routing protocol developed by ABB. It is specially designed for use in automation networks with limited topology. (ABB Inc. 2012a.)

3.2 Operations software

End user configuration and programming of the DCS is done by the help of supplier developed applications. This section explains functions of different applications. Because the experimental part of this work was done with the ABB System 800xA DCS, its applications will also be used as a reference. The main configuration tools, which also were utilized to complete this work, are the Engineering and Operator Workplaces, Control Builder M Professional and the Process Graphics Editor.

3.2.1 ABB System 800xA

The Industrial IT Extended Automation System 800xA was first released in December 2003. It is the main automation system in use at Freeport Cobalt Ltd. and very popular in other process industry plants as well. Specifically, the System 800xA incorporates process control, production management, safety, discrete logic and sequence control, advanced control, information management, smart instrumentation, smart drives and MCCs, asset management, and document management capabilities in a singular virtual database environment. (ABB Inc. 2013.)

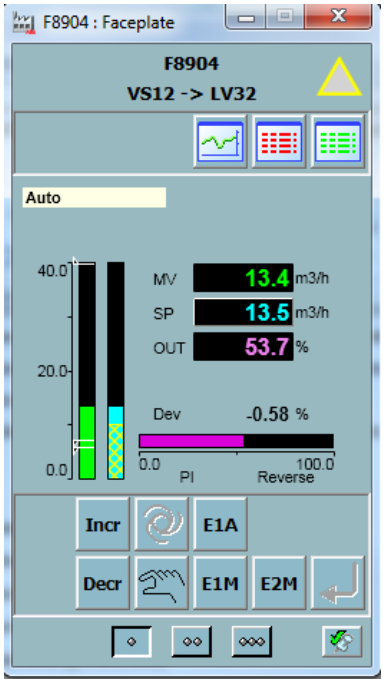
3.2.2 Operator Workplace

The production environment represents the current approved state of the system. The state consist of the solution running in the controller and the graphics, alarm lists, and trend displays. used by the production system. This is the environment the operators use to monitor and interact with the plant processes. The processes are displayed and interacted with in graphical displays. (ABB Inc. 2012b.)

Display hierarchy is usually divided into two to five levels to show different amounts of detail. In the first level is the overview of the respective plant section. It is a main menu-like screen where the operator can navigate into specific units. In the unit screen, the operator can see the process flow in a logical order that corresponds with the physical process environment. This is the screen used in Graph 6. In the third level is the

information related to a specific interactive object in the unit screen. For example, the operator can open text documents which explain the function of group starts, motors, and valves. (ABB Inc. 2012b.)

By clicking on a dynamic object on the process screen, a faceplate appears. The faceplate shows detailed information of the selected object, such as the current measured process value and its range and alarm limits. Operators can also manually control the selected object from the faceplate by setting the device on manual control, if allowed in the programming. A faceplate of a control loop can be seen in Graph 8. The bottom buttons are used to select a control method: automatic, manual or external control. The abbreviations are MV = Manipulated Variable, SP = SetPoint, Dev = Deviation, and PI = Proportional Integral. More detailed information about control loops can be found in section 4.1.2.



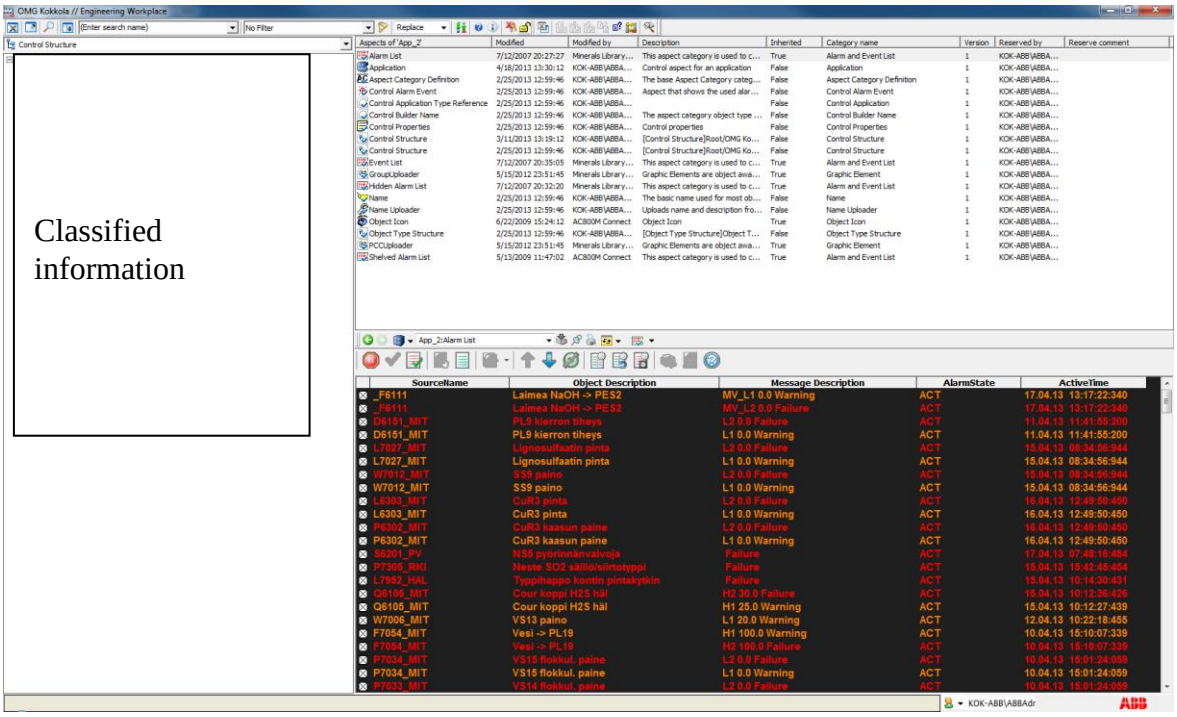
GRAPH 8. PI Control loop faceplate

Navigation between different menus is usually done by tabbed navigation. Tabs are located at the top of the screen and provide quick access to important graphics. The tabs at Freeport Cobalt are quick links to different plant sections, such as refining and chemicals production. One could think of these as his favorite sites on an internet browser: this makes it easy for a new user to relate to the operating system.

3.2.3 Engineering Workplace

The Plant Explorer, or Engineering Workplace, is used to create, delete, and organize aspect objects and aspect within the 800xA System. It organizes the aspect objects in structures according to functionality, and location. Plant Explorer is the main tool used by engineers for exploring and building hierarchically structured models of a plant or system. The structures represent different views of the plant. The following are examples of different types of structures. Functional Structure displays the plant from the process perspective. It is an overview of the functionality of items in the plant. It is used for operation of the plant. Location Structure displays the physical layout of where equipment is located in the plant. It is primarily used for maintenance tasks. Control Structure displays the control network in terms of networks, nodes, fieldbuses and stations. (ABB Inc. 2012b.)

All the entities included in a plant are represented as objects; for example, valves, motors, controllers and tanks. These objects have relevant information stored in aspects, as shown in Graph 9. This information is, for example, process graphics, control dialogs and alarm pages. In Graph 9, the aspect objects are in the left column and a list of the aspects connected to it is in the right column. Below it, the screen shows details of the selected aspect, in this case an alarm and event list. The aspects are stored in, and run from the aspect servers, as explained in section 3.1.2. (ABB Inc. 2012b.)



GRAPH 9. Engineering Workplace screen

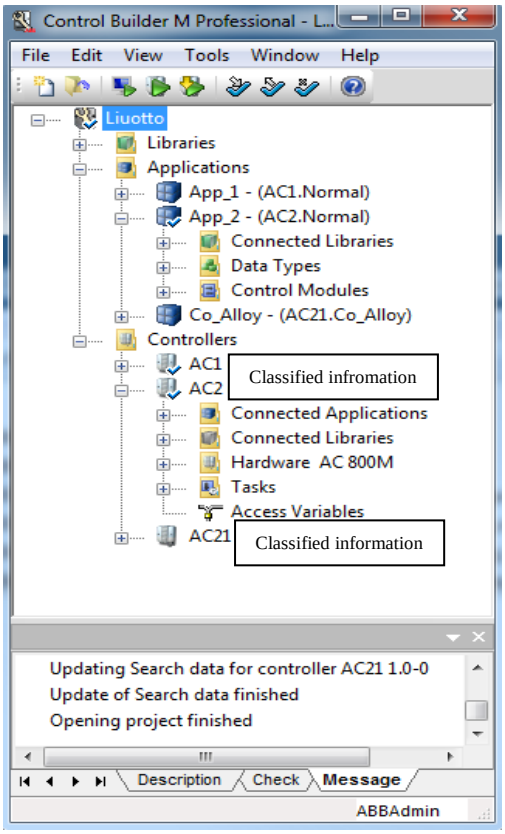
3.2.4 Control Builder M Professional

All the program code and I/O definitions which were made was with the help of the Control Builder. Control Builder is a programming tool that contains programming editors, a compiler, standard libraries for developing controller applications and standard hardware types in libraries for configuring the controllers. The Control Builder also includes system firmware and common functions such as control system templates, task supervision, security and access management. (ABB Inc. 2012d.)

At Freeport Cobalt, the code is divided in several different project folders by the geographical placement of the controllers. The one that was worked with was called “Liuotto” which is the raw material refinery of the plant. The project breaks down in a tree formation similar to Windows Explorer. The tree also represents the hierarchical structure of the project from top to bottom. Details of the programming sequence can be found in section 4.1. The project view screen can be seen in Graph 10.

The project contains three main folders: Libraries contains a storage of reusable programming elements that can be used In the program code and hardware definitions.

Applications contain the device tag definitions and program code. At Freeport Cobalt there is one application for each controller. Controllers folder contains hardware definitions. i.e. the setup of the controller rack, I/O connections and the controller tasks which define the priority and interval time in program execution.



GRAPH 10. Liuotto -project in Control Builder

3.2.5 Graphics Builder

Graphics Builder is a tool that enables configuration of graphic aspects. Graphic aspects are graphic displays, graphic elements and faceplates. The configuration of graphic displays is easy and intuitive; hence this does not require lot of programming skill.

At Freeport Cobalt the displays are usually made to correspond with the P&I-diagrams. Tanks and flow lines are made into static objects, whereas the devices are presented in dynamic objects which can be interacted with. (ABB Inc. 2012c.)

There are several features included in the Graphics Builder: Expression Builder, which allows assigning of expressions, which is to specify the relationship between the process data and the data that will be displayed to the operator. Element Browser contains ready-made objects from where the user can select appropriate graphic elements for inclusion. Design and test function permits building of graphic aspects in design mode and then to check their behavior in test mode. Finally, a Solution Library where user-defined graphic solutions can be stored for reuse. (ABB Inc. 2012c.)

3.3 Programming languages

In this section, three IEC 61131-3 programming languages used in the practical work of this thesis are examined. The main language used in the ABB System 800xA at Freeport Cobalt Ltd. is the Structured Text. As explained in the introduction, there is a second, older, DCS still in use, the ABB Advant. Advant was programmed entirely in a type of Function Block Diagram. The reason for the switch of programming language into ST is the more universal structure of the language. This allows individuals from wider educational backgrounds to be able to understand and program the system more efficiently. The third language, the Sequential Function Chart, is used at Freeport Cobalt in digressive basis. It is mainly used to group start/stop field devices in a chain. This language, however, is being replaced by the System 800xA's integrated Group Sequence-function (Section 4.2.3), which is very quick to use and easy to learn because it is specifically designed for group starts/stops, whereas the SFC can be used in many other sequential process operations. (Korpijärvi 2013.)

Table 1. Examples of basic language functions (ABB Inc. 2012e.)

Assignment functions	*(move, :=)*
Boolean functions	*(not, and, &, xor, or, >=)*
Arithmetic functions	*(expt, mul, div, add, sub, mod, abs)*
Relational functions	*(<, >, <=, >=, =, < >)*
Conditional functions	*(if, then, else)*
Iteration functions	*(for, while, repeat)*

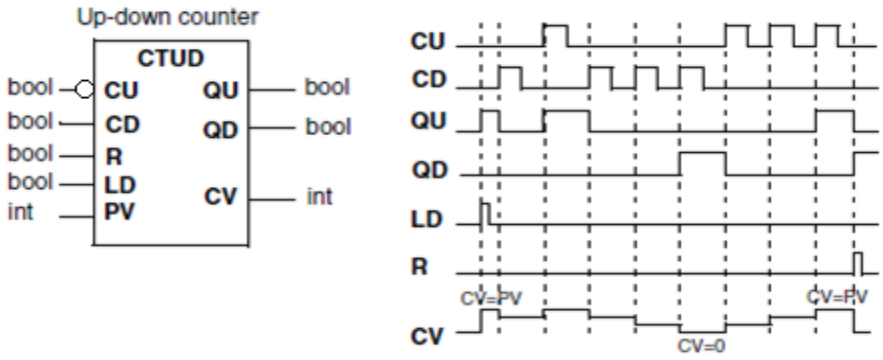
Further details of the three languages used and studied can be found in the next three subsections. The main language used in this work was the Structured Text. Function Block Diagram was also used in some extent. This language was mainly used in the older ABB Advant DCS. (Paloranta, M 2013.)

3.3.1 Function Block Diagram

Function Block Diagram (FBD) is a high-level graphical programming language in which the control functions are divided into a number of function blocks or functions connected by flow signals. A function block may contain simple logical conditions, timers or counters, but they can also provide a complex control function to a subprocess in a machine or even an industrial plant. (ABB Inc. 2012e.)

FBD is very suitable for describing Boolean logic with associated timers, counters and bistables. Most programmable controllers have such function blocks predefined in standard libraries, such as discussed in section 3.2, for direct use by the programmer. In a Function Block Diagram editor, the parameters of functions and function blocks are shown as pins on the boxes. The assignment of values to parameters is shown by lines connecting the pins. An example of a FBD program can be seen in Appendix 11. (ABB Inc. 2012e.)

Graph 11. is an example of an FBD up-down counter with a negated Boolean signal to CU. The block is used for counting up and counting down the value in output CV. Integer data type is needed to present other values than 1 or 0 (Boolean values). The diagram on the right of the block is a timing diagram representing its function, horizontal axis representing time. (ABB Inc. 2012e.)



GRAPH 11. Example of a CTUD counter block (ABB Inc. 2012e.)

The downside of FBD is its weak support for conditional statements when one or more actions should be reiterated for a specific number of times, or only as long as a certain condition is fulfilled. This kind of construct is much easier to accomplish in the ST language with statements such as FOR, WHILE, or IF. (ABB Inc. 2012e.)

3.3.2 Structured Text

Structured Text (ST) is a high level programming language, similar to Pascal and C, that has been specifically designed for use in programmable controllers. It is compact, highly structured and contains a comprehensive range of constructs for assignments, function/function block calls, expressions, conditional statement, iterations and more. This language was used to code the majority of the practical work. The code is simple to write and easy to read, because of its logical and structured layout. (ABB Inc. 2012e.)

On the whole, ST is very effective tool for developing control applications. The language is a good general purpose tool for expressing different types of behavior with all kinds of structured variables. This should be the choice of language when iterations are needed, that is when something is to be repeated a fixed or conditional number of times.
The ST language contains a list of statements, such as assignment statements like `(variable := expression)`, conditional statements (`if, then, else`), iteration statements `(for, while, repeat)` and control statements `(exit, return)`. (ABB Inc. 2012e.)

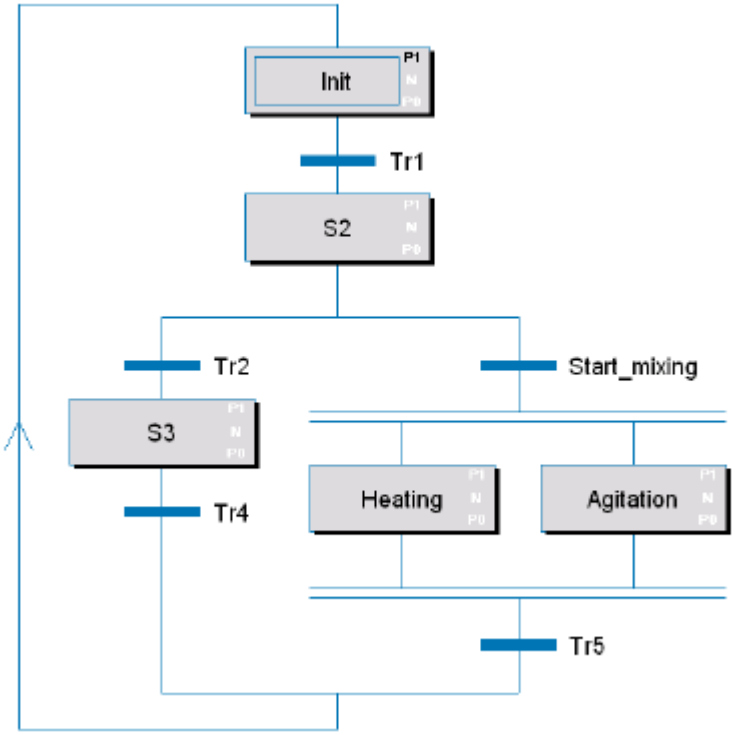
The following example shows when Function Blocks, as the one in Graph 11. are called by a statement consisting of the function block name followed by a list of named input and output parameter value assignments. The output is written in to a variable for use in other parts of the application.

```
PulseTimer1     ( PulseTime := Time1s,
                  Enable := true,
                  PeriodTime := Time20s,
                  Out => W7302_Value ):
```

3.3.3 Sequential Function Chart

The Sequential Function Chart (SFC) programming language allows the user to describe the sequential behavior of the control program graphically. The language was studied mainly to interpret sequences previously made in the plant. SFC concept enables all control actions for a process to be described in one compound sequence structure. Sequences can be hierarchical, that is, action chains can be grouped to give a clear, high-level presentation of the process control unit. (ABB Inc. 2012e.)

Graphically, SFC is visualized in the editor in a series of steps represented by rectangular boxes and connected by vertical lines. Each step represents a physical state of the system being controlled. On each connecting line, there is a horizontal bar representing a transition. The transition is associated with a transition condition which, when true, deactivates the step before the transition and activates the step after the transition. An example of this can be seen in Graph 12, where Tr1,2,3.. are the transitions. (ABB Inc. 2012e.)



GRAPH 12. A sequence structure. (ABB Inc. 2012e.)

Even though SFC has many advantages as a design and structuring tool, it is not a complete programming language. Therefore, the transition conditions and action descriptions have to be programmed with one or more of the other available programming languages. In Graph 12. for example, there is code written inside the *Init* box. At Freeport Cobalt, the language can be either ST or FBD. (ABB Inc. 2012e.)

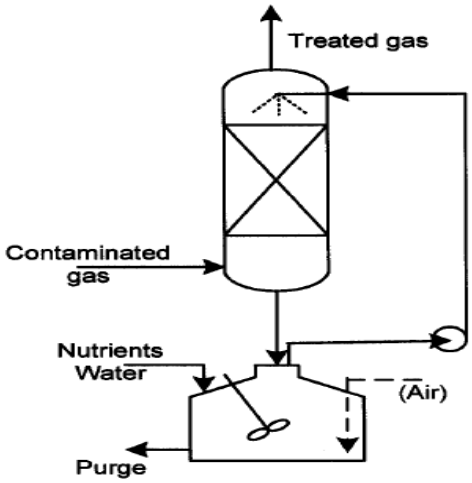
3.4 Summary

This chapter covered the aspects which were needed to be studied and internalized in order to understand automation in process industry as a whole. Architecture and DCS structure was studied as well as related devices and networking methods. Insights to the programming tools were given and their practical purpose discussed. Finally, some of the IEC 61131-3 programming languages were looked into and their advantages and disadvantages resolved. Throughout this paragraph, references to Freeport Cobalt Ltd.'s use cases were given in most of the examined subjects, for purpose of relation to real-world cases.

4 AUTOMATION OF H_2S DESULPHURIZATION SYSTEM

The first part of the practical work of this thesis was to design, develop and reprogram the automation of the hydrogen sulfide desulphurization system. Hydrogen sulfide is an explosive, heavier than air, poisonous and corrosive gas. At Freeport Cobalt Ltd. the gas is used in precipitation of metals in various stages of solution purification. The residue gas is collected from different stages and directed by the means of vacuum suction to the desulphurization system, from here on referred to as the scrubber. In the scrubber the toxic H_2S is sprayed with a mixture of water and sodium hydroxide to absorb the sulfur to that purification solution. The purified gas is then released via the chimney, in which the scrubber is located, to the open air. The operation of the scrubber is visualized in Graph 13. (Paloranta 2013.)

The treatment of the gas is very important due to its high toxicity. Upon inhalation, hydrogen sulfide reacts with enzymes in the bloodstream and inhibits cellular respiration resulting in pulmonary paralysis, sudden collapse, and death. Continuous exposure to low (15-50 ppm) concentrations will generally cause irritation to mucous membranes and may also cause headaches, dizziness, and nausea. Higher concentrations (200-300 ppm) may result in respiratory arrest leading to coma and unconsciousness. Exposures for more than 30 minutes at concentrations greater than 700 ppm have been fatal. (M. Syed, G. Soreanu, P. Falletta and M. Béland 2006.)



GRAPH 13. A bioscrubber system for removal of H_2S (Syed, M. Soreanu, G. Falletta P. and Béland, M. 2006.)

4.1 Operation principle

For the efficient operation and monitoring of the system, a set of devices are needed and are to be automated. Each device is connected to, and controlled by the DCS. To explain the operation of the system devices, they will be referred to with the help of device tags found in Appendix 1.

The pump M6138 creates a vacuum in the pipeline and sucks the contaminated gas into the scrubber tower. This pump can be seen in Graph 14. labeled as 21PUH18. The pump is controlled by the PID controller tagged P6109, additional examination of this in section 4.1.2. Once in the scrubber, the gas is sprayed with the cleansing solution of water and sodium hydroxide that is pumped from a container in the bottom of the scrubber. The circulation pump is tagged M6139. Some of the cleansing solution is evaporated with the gas and must be constantly refilled. Water is pumped in to the container with a constant rate of approximately 300 l/h. To get the right solution, sodium hydroxide ratio in the water must be suitable. The sodium hydroxide flow in to the container is kept at the correct amount with the help of the PID controller tagged F6111; this is also more closely examined in section 4.1.2. The treated gas is then let out in to the open air.

In addition to the control loops, measurements which do not directly control anything are also needed. These measurements allow closer monitoring of the system operation that facilitates making the system more efficient. Also alarms are triggered by the measurements programmer-set high-limit, or low-limit values. The alarms alert operators of abnormal conditions so that appropriate actions can be performed.

GRAPH 14. Indoor view of the desulphurization system

4.1.1 Instrumentation

Instruments are all the devices connected to the DCS, excluding electric motors. Table 2. is a list of all the instruments used in the desulphurization system. The table shows I/O connections of each instrument, both analog and digital, tags, descriptions and measurement properties. Columns labeled with 'ex' are connections to and from ATEX-certified I/O modules. Also the instruments themselves must be ATEX-certified due to the explosive nature of hydrogen sulphide.

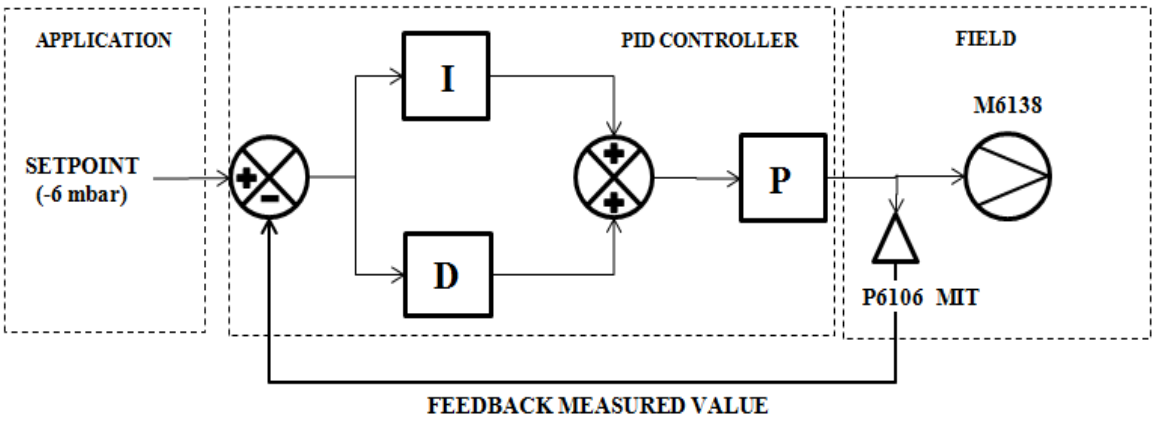
TABLE 2. Instrument list of the H_2S desulphurization system

Tag	Description	AI	DO	AI ex	AO ex	DOex	Range	Unit
F6109	Water -> PES2	1					0 - 1000	l/h
F6110	PES2 cycle	1					0 - 15	m3/h
F6111	NaOH -> PES2	1					0 - 200	l/h
H6132	H2S pressure release valve		1			2	on/off	
H6133	Emergency shower, PES2 area					1	on/off	
L6112	21PES2 level			1			0 - 100	%
P6109	21PES2 gas vacuum pressure control			1			-50..0	mbar
P6110	21PES2 pressure-difference			1			0 - 50	mbar
Q6106	PES2 area H2S alarm			1			0 - 50	ppm
Q6107	PES2 pH			1			0 - 14	pH
Q6108	PES2 cycle pH			1	1		5 - 14	pH
Q6110	PES2 H2S measurement	1					0 - 500	ppm
T6111	PES2 temperature			1			0 - 150	°C
T6112	PES2 H2S pipeline 1 temperature			1			0 - 200	°C
T6113	PES2 H2S pipeline 2 temperature	1					0 - 200	°C
T6114	PES2 area temperature			1			-50..+50	°C

4.1.2 Control loops and control theory

Control loops are used to automatically regulate or direct a device to a desired setpoint. They are applied when a process quantity is sensitive to deviation due to changing environmental conditions or anomalies. In practice, the operator sets the desired setpoint in the control loop faceplate, see Graph 8. on page 15, and selects the desired control method: automatic, manual or external control.

As expressed in section 4.1, a PID controller must be used with the pump M6138. The pressure in the pipeline must be kept at -6.0 mbar for efficient gas removal. The pressure might deviate due to different factors such as dirt in the pipeline. Because of this, the suction force must be constantly adjusted. This is achieved by the PID controller that sets the revolution speed for the pump. As shown in Graph 15. a pressure sensor is installed to the main transfer line, tagged P6109_MIT that gives feedback to the PID controller, tagged P6109, creating a closed control loop. If there is deviation between the setpoint and feedback measured value, the controller will compensate for it.



GRAPH 15. P6109 closed control loop

Inside the PID controller, three tuning parameters exist. The proportional gain term sets a constant output value in relation to the deviation between the MV and SP. The Integral gain accelerates the output value if the feedback does not change towards the desired SP fast enough. The derivative term examines the rate of change from a deviated value towards the SP and tries to compensate for deviation before it occurs.

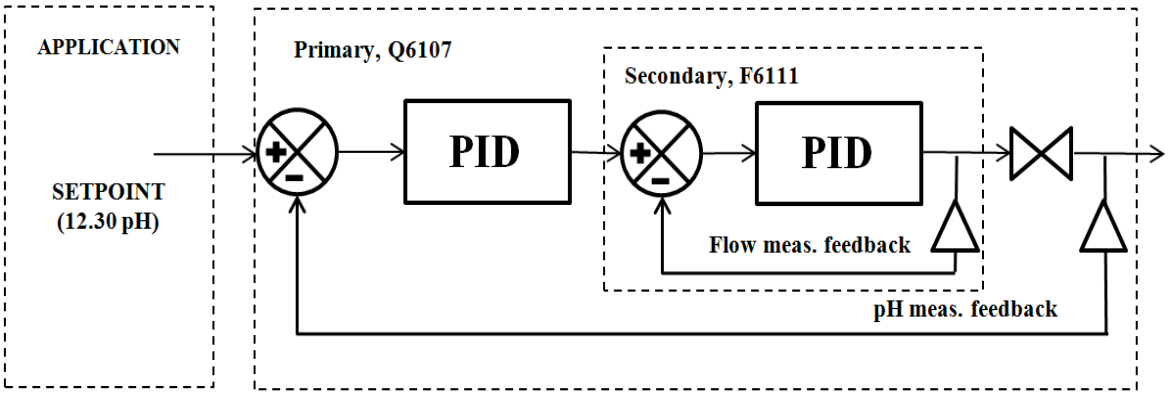
Tuning the parameters can be done either manually trying different parameters (trial and error), or by mathematically calculating the behavior of the system. By introducing a term G that stands for the pump's dynamics, in front of the P -term in Graph 15, the response of the control event can be estimated with the help of the transfer function G(s). In the formula (4.1), Pv stands for Process Value, and Sp for Setpoint.

$$\begin{cases} Pv = Sp(I + D)PG - PvPG(I + D) \\ Pv\big(1 + PG(I + D)\big) = Sp(I + D)PG \end{cases} \tag{4.1}$$

$$\frac{Pv}{Sp} = \frac{PG(I + D)}{1 + PG(I + D)} = G(s) \tag{4.2}$$

An additional PID controller is needed for the regulation of the correct mixture of sodium hydroxide. This is done by measuring the pH and flow values of the solution circulating in the tower, these values work as separate feedbacks for two controllers. The controllers steer a control valve letting sodium hydroxide flow to the tank in the right amount.

The sodium hydroxide pressure is constant and susceptible to deviation due to its usage in other processes as well, therefore, a more stable control method was needed. Two PID controllers were applied, one measuring the pH value and one the flow of sodium hydroxide. As shown in Graph 16, the controller measuring the pH value is tagged Q6107 (primary). Its output is connected to the setpoint input of the other controller tagged F6111(secondary) that measures the flow rate. F6111 output finally guides the control valve in order to achieve the appropriate flow. This control method is called cascade control. The stability of the control system with this method is increased, because disturbances from rapid changes of the secondary controller will not affect the primary controller.



GRAPH 16. Q6107 and F6111 cascade control

Also in this case, calculating the transfer function can be helpful in order to estimate control behavior. To ease the calculation, the secondary controller transfer function is first resolved. The secondary transfer function is then placed in the main equation where $G_2(s)$ is the secondary function and G_v is the control valve dynamics. In order to narrow down the equation, the controller terms, P, I and D, are abbreviated in a single block: PID. First, the transfer function for the secondary PID controller is resolved in (4.3) and (4.4). Here there is no G term due to the fact that there is no actuator directly controlled by this control loop.

$$\begin{cases} Pv = SpPID - PvPID \\ Pv(1 + PID) = SpPID \end{cases} \tag{4.3}$$

$$\frac{Pv}{Sp} = \frac{PID}{1 + PID} = G_2(s) \tag{4.4}$$

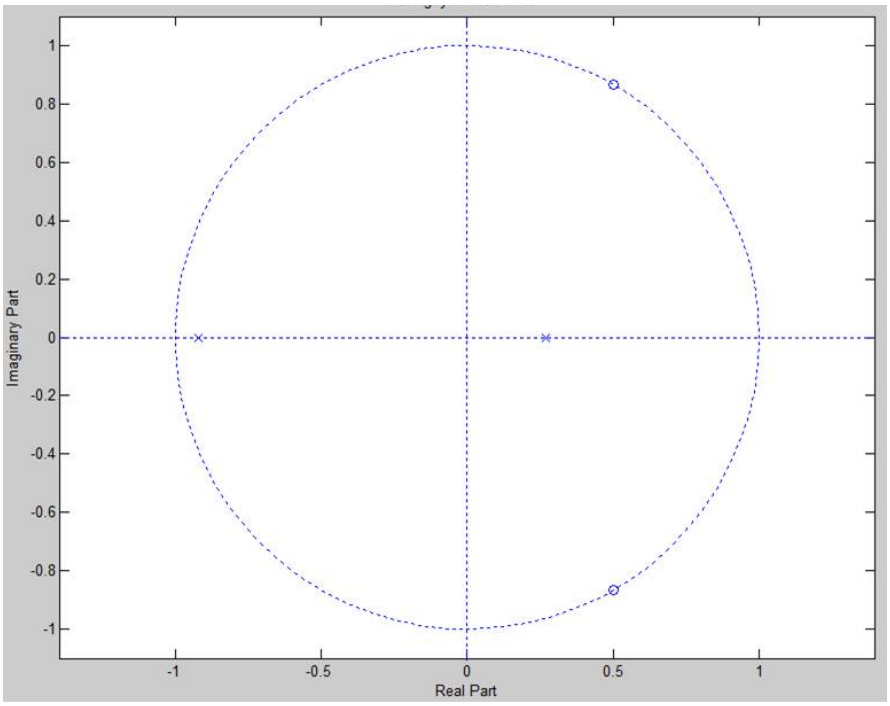
After the secondary controller transfer function $G_2(s)$ is resolved, it can be inserted in to the main equation for the primary controller loop. Here also the valve dynamics G_v is taken into account.

$$\begin{cases} Pv = SpPID_2G_2(s)G_v - PvPID_2G_2(s)G_v \\ Pv(1 + PID_2G_2(s)G_v) = SpPID_2G_2(s)G_v \end{cases} \tag{4.5}$$

Finally by expanding the final equation, it results as the transfer function for the whole cascade control event G(s).

$$\frac{Pv}{Sp} = \frac{PID_2 G_2(s) G_v}{1 + PID_2 G_2(s) G_v} = \frac{PID_2 \ \frac{PID}{1+PID} G_v}{1 + PID_2 \ \frac{PID}{1+PID} G_v} = G(s) \qquad (4.6)$$

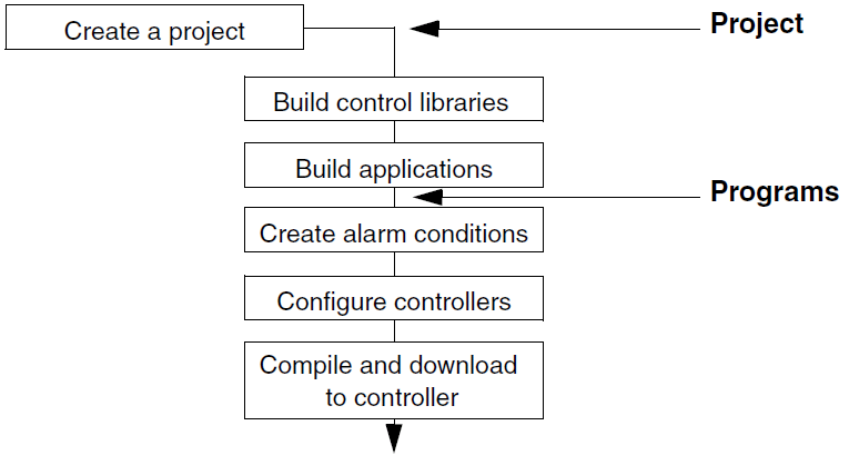
Once the complete transfer function is resolved, values can be inserted in order to study the stability of the system. When calculated, the result is in two parts, the poles and the zero. These values can then be inserted into a pole-zero plot, which is a graphical representation of the transfer function. With this plot, one can determine the stability of the system in a control event. See Graph 17. for an example of a pole-zero plot. For example, if the poles are a imaginary result on the negative real axis, it can be said that the system is stable, but susceptible to oscillation.



GRAPH 17. Example pole-zero plot

4.2 Practical programming phases

The programming is done in the same order as the hierarchical structure of each project. It starts with adding a new project folder, which in this case was already made and designated as the Liuotto project. After this, appropriate control libraries are connected to the project. As the Liuotto project was already made, these were also readily connected. The next phase was where programming work was initiated. Graph 18. illustrates the steps from creating a new project to download.

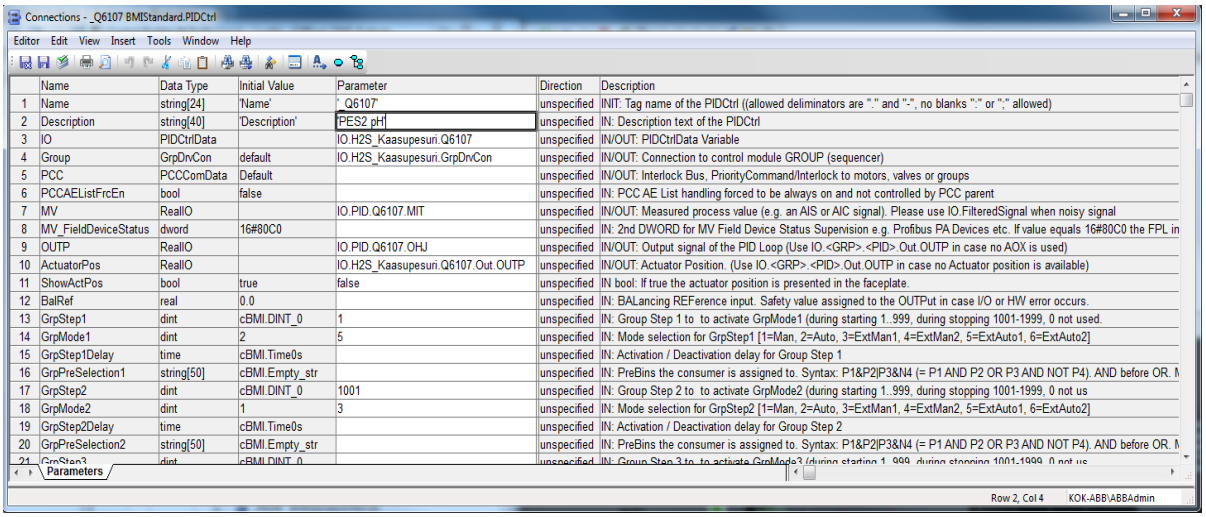


GRAPH 18. Steps for building a project in ABB 800xA

The practical programming was started by adding a new application labeled App_2 in the Control builder (see section 3.2.4, Graph 10.). The next step was to connect the standard libraries to the application, (about libraries see sections 3.2 and 3.3.1). The libraries contain, inter alia, definitions for different data types which were added next. In the data type folders, instrument and device tags were added so that the program would know what kind of device each tag was. For example, a temperature sensor has a single current loop connection that sends information to the DCS; therefore it is of AI (Analog Input) data type. Once all the devices (Table 2. on page 26) and motors were added with their respective data types and tags, it was time to parameter them.

All the parameter definitions for a device contained in a control module. Common parameters for each control module were tag names and descriptions, connection to the respective data variable and alarm class. Rest of the parameters varied by data type, for example, a PID controller control module contained 126 parameter definitions. These

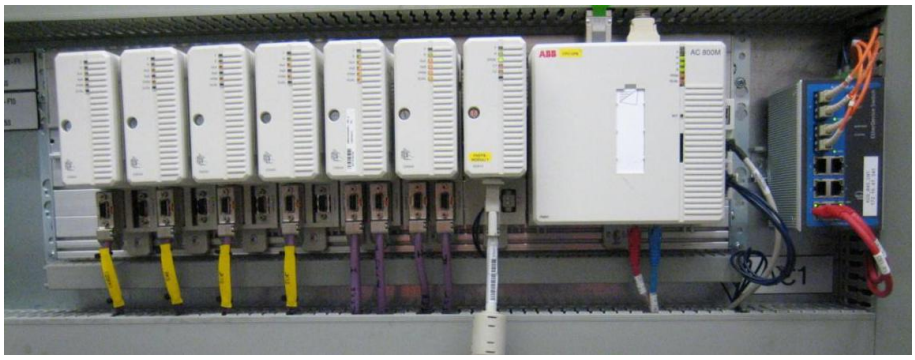
parameters define alarm limits, tuning parameters, enabling of various control modes and more. All of the parameters were not used and the system had predefined default values for those left empty. In Graph 19. are some of the parameters of the PID Q6107. On row 9 is the output connection of the PID. In case of the PID P6109, there was written the connection to the pump M6138 input. The last step of defining the devices to the system was the configuring of hardware and physical I/O connections.



	Name	Data Type	Initial Value	Parameter	Direction	Description
1	Name	string[24]	'Name'	'_Q6107'	unspecified	INIT: Tag name of the PIDCtrl ((allowed deliminators are "." and "-", no blanks ":" or ";" allowed)
2	Description	string[40]	'Description'	'PES2 pH'	unspecified	IN: Description text of the PIDCtrl
3	IO	PIDCtrlData		IO.H2S_Kaasupesuri.Q6107	unspecified	IN/OUT: PIDCtrlData Variable
4	Group	GrpDrvCon	default	IO.H2S_Kaasupesuri.GrpDrvCon	unspecified	IN/OUT: Connection to control module GROUP (sequencer)
5	PCC	PCCComData	Default		unspecified	IN/OUT: Interlock Bus, PriorityCommand/Interlock to motors, valves or groups
6	PCCAEListFrcEn	bool	false		unspecified	IN: PCC AE List handling forced to be always on and not controlled by PCC parent
7	MV	RealIO		IO.PID.Q6107.MIT	unspecified	IN/OUT: Measured process value (e.g. an AIS or AIC signal). Please use IO.FilteredSignal when noisy signal
8	MV_FieldDeviceStatus	dword	16#80C0		unspecified	IN: 2nd DWORD for MV Field Device Status Supervision e.g. Profibus PA Devices etc. If value equals 16#80C0 the FPL in
9	OUTP	RealIO		IO.PID.Q6107.OHJ	unspecified	IN/OUT: Output signal of the PID Loop (Use IO.<GRP>.<PID>.Out.OUTP in case no AOX is used)
10	ActuatorPos	RealIO		IO.H2S_Kaasupesuri.Q6107.Out.OUTP	unspecified	IN/OUT: Actuator Position. (Use IO.<GRP>.<PID>.Out.OUTP in case no Actuator position is available)
11	ShowActPos	bool	true	false	unspecified	IN bool: If true the actuator position is presented in the faceplate.
12	BalRef	real	0.0		unspecified	IN: BALancing REFerence input. Safety value assigned to the OUTPut in case I/O or HW error occurs.
13	GrpStep1	dint	cBMI.DINT_0	1	unspecified	IN: Group Step 1 to to activate GrpMode1 (during starting 1..999, during stopping 1001-1999, 0 not used.
14	GrpMode1	dint	2	5	unspecified	IN: Mode selection for GrpStep1 [1=Man, 2=Auto, 3=ExtMan1, 4=ExtMan2, 5=ExtAuto1, 6=ExtAuto2]
15	GrpStep1Delay	time	cBMI.Time0s		unspecified	IN: Activation / Deactivation delay for Group Step 1
16	GrpPreSelection1	string[50]	cBMI.Empty_str		unspecified	IN: PreBins the consumer is assigned to. Syntax: P1&P2|P3&N4 (= P1 AND P2 OR P3 AND NOT P4). AND before OR. M
17	GrpStep2	dint	cBMI.DINT_0	1001	unspecified	IN: Group Step 2 to to activate GrpMode2 (during starting 1..999, during stopping 1001-1999, 0 not us
18	GrpMode2	dint	1	3	unspecified	IN: Mode selection for GrpStep2 [1=Man, 2=Auto, 3=ExtMan1, 4=ExtMan2, 5=ExtAuto1, 6=ExtAuto2]
19	GrpStep2Delay	time	cBMI.Time0s		unspecified	IN: Activation / Deactivation delay for Group Step 2
20	GrpPreSelection2	string[50]	cBMI.Empty_str		unspecified	IN: PreBins the consumer is assigned to. Syntax: P1&P2|P3&N4 (= P1 AND P2 OR P3 AND NOT P4). AND before OR. M
21	GrpStep3	dint	cBMI.DINT_0		unspecified	IN: Group Step 3 to to activate GrpMode3 (during starting 1..999, during stopping 1001-1999, 0 not us

GRAPH 19. Control module parameters of PID controller Q6107

The hardware is modelled at the bottom of the Control Builder screen. The controller that was worked with was labeled AC2. In the folder AC2 was the application folder, libraries folder, hardware folder and tasks folder. The application folder tells the controller which application(s) it will be running. In this case App_2, which was previously created, was connected. Libraries folder contain a set of predefined hardware types stored in standard hardware libraries. These libraries contain definitions for devices such as controllers, I/O modules, CI modules, and more. Hardware folder models the actual hardware that is connected to the controller. The folder opens a hierarchical view first of the controller rack (Graph 20.) and under each CI module are the connected I/O clusters (Graph 21.) to which the actual instruments are connected.



GRAPH 20. A controller rack with several CI modules connected

GRAPH 21. An I/O cluster with redundant fieldbus slave units

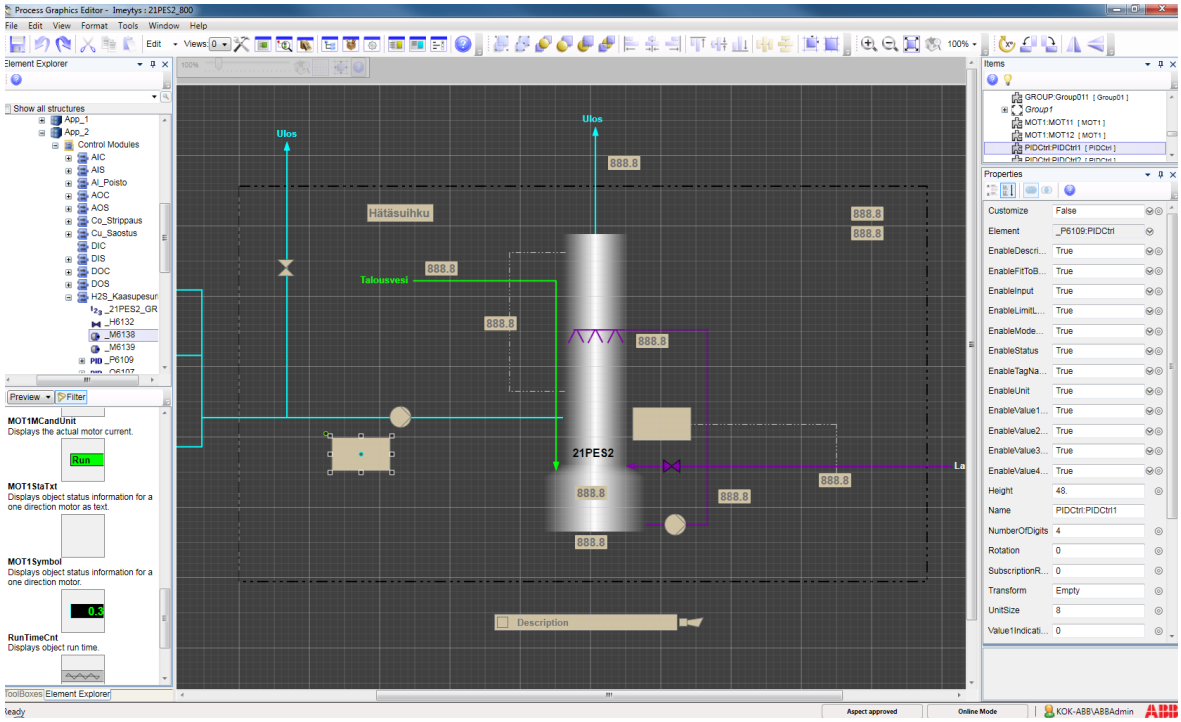
Meanwhile the researcher was designing the automation of the system, other designers drew and documented the instrument I/O connections in circuit diagrams, which were used to look up the hardware setup. An example of one these circuits can be found in Appendix 5. Once elucidated to which I/O port each instrument will be connected, each port of the I/O modules were configured with the appropriate variables and their properties. The variables are written in the following way: *application reference.data type reference.data type.device tag.signal type.* For example: `App_2.IO.AIS.T6113_MIT.Signal`.

4.2.1 Process graphics display

The code would be almost useless without the process graphics whereby the operators can interact with the desulphurization system. The creation of the process graphics was started by creating a new aspect in the engineering workplace (Section 3.2.3, Graph 9.). Even though the graphic aspect is situated on the aspect server (Section 3.1.2) the same way as other graphics, the unfinished graphics did not display to the operators because there was

no way to navigate to the created aspect from the operator environment (Section 3.2.2). The link was created when it was time to introduce the finished system.

Configuring of the graphics was done with the Graphics Builder (Section 3.2.5). The first step was to create the dynamic graphic elements of the system devices. This was done by browsing the elements in the Element Explorer inside the Graphics Builder. The Element Explorer is directly connected to the Engineering Workplace control structure. Here the previously created control modules can be seen and picked into the graphics as dynamic elements. The elements are then placed in the graphics in the spot corresponding to their actual placement. In Graph 22. the dynamic objects can be seen as light brown boxes. These objects can be later interacted with by the operators. Static objects, which cannot be interacted with, are the pipelines and the scrubber tower.



GRAPH 22. Configuration of H_2S desuplhurization system graphics

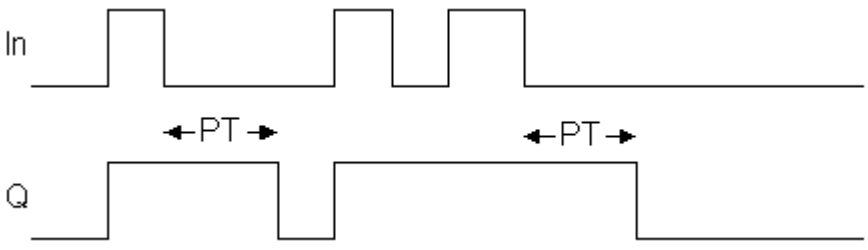
After placing the objects, their representation to the operator was also defined. For example, the system does not distinguish motor types before their representation is defined in the graphics. In this case the motors were pumps, so their symbols were changed to pump symbols instead of conventional motors. Also definitions for alarm representation were made for the emergency shower activation. A bar was also added for the group

sequence start of the system, explained further in paragraph 4.2.4. Documents were also created to explain the function of interlocks and the group sequence start bar. The final revision of the graphics screen can be seen in Appendix 3.

4.2.2 Interlocks

An interlock is introduced in conditions where a device is not allowed to be started, shut down or operated before certain conditions are met. Interlocks are mainly used for enforcing safe operation and durability of system devices. In the scrubber, one important interlock was needed for the valve tagged H6132. The valve operates as a release valve in case the vacuum of the pipeline fails. This failure might occur for example if the pump M6138 ceases operating. If there was no release valve, the toxic gases would build up indoors. The valve was programmed to open and let the gases straight outdoors without going through the scrubber when the pipeline pressure reaches a high limit of -2.0mbar. The valve does not automatically close once opened. For the operator to manually close the valve again, the pressure must be within the acceptable range between -49.0 and -4.1mbar.

The code was written in ST under the H_2S scrubber folder. First, a timer function block was introduced for the purpose of hysteresis. This means that if a sudden spike, like a signal interference, occurs the valve will not unnecessarily open. The timer creates a 10 second delay before the valve is opened and interlock put in to effect. Graph 23. shows the behavior of the block when an input signal is introduced. The following code is a conditional statement that defines that the interlock was to be put in to effect after the timer block gives an output signal, if not, the interlock was not to be in effect.



GRAPH 23. TOf Function Block timing diagram

```
H6132_TOff      ( In := not IO.H2S_Kaasupesuri.P6109.Out.MV_H2.Value,
                  PT := Time10s );

if H6132_TOff.Q then;
   IO.H2S_Kaasupesuri.H6132.In.IB1 := true;
else
   IO.H2S_Kaasupesuri.H6132.In.IB1 := false;
end_if;
```

Another interlock was needed for the pump tagged M6139. The pump circulates the fluid from the tank located at the bottom of the scrubber to the showers spraying the gas. The interlock triggers when the fluid level in the tank, indicated by measurement L6112_MIT, reaches the limit L2, set at 10%. This interlock was programmed because the pump can sustain severe damage if it operates without fluid running through it. The code is similar to the one used previously:

```
M6139_TOff      ( In := not IO.AIS.L6112_MIT.L2.Value,
                  PT := Time10s );

if M6139_TOff.Q then;
   IO.H2S_Kaasupesuri.M6139.In.IB1 := true;
else
   IO.H2S_Kaasupesuri.M6139.In.IB1 := false;
end_if;
```

A document explaining these events was created and placed as an aspect in the engineering workplace. A link to this aspect was then created in the graphics screen as a button. Documentation is important to be always available to give operators an overview of how the system works and in respect prevent confusing situations from occurring. The interlock document can be seen in Appendix 6.

4.2.3 Cascade control connection

This section examines the programming method that was used in order to achieve the cascade control studied in section 4.1.2. Two separate PID control modules were created and parameterized in the Control Builder. The PID's were then added to the process

graphics where the cascade connection was indicated by a dotted line between the two elements. It still had to be defined that the actual setpoint of the controller F6111 is set from the output of controller Q6107. This definition had to be programmed in ST language in the Control Builder as shown below:

```
(* Q6107 -> F6111 *)
IO.H2S_Kaasupesuri.F6111.In.ExtRefAuto1 :=
(IO.PID.F6111.MIT.Parameters.Max - IO.PID.F6111.MIT.Parameters.Min) *
 IO.PID.Q6107.OHJ.Value/100.0 + IO.PID.F6111.MIT.Parameters.Min;
```

The first row of the code is to connect the output value of the PID Q6107 to the external reference setpoint of controller F6111. However, in order to do this, the value fed into the external reference must be scaled from the percentage output of Q6107 to match the value in the measurement range of the controller F6111. For example, if the measurement minimum and maximum values of the secondary controller are 120 and 200, it means that the range is 80 (200 – 120 = 80). When the output of the primary controller is at 10%, this multiplied with the range value of the secondary controller results in 8 (10% * 80 = 8). Finally, the result is added to the minimum range value of the secondary controller to get the actual setpoint value (120 + 8 = 128). The formula for the scaling resulted as the following below:

$$Setp = \left[(Pmax - Pmin)\left(\frac{Outp}{100}\right)\right] + Pmin \tag{4.7}$$

where

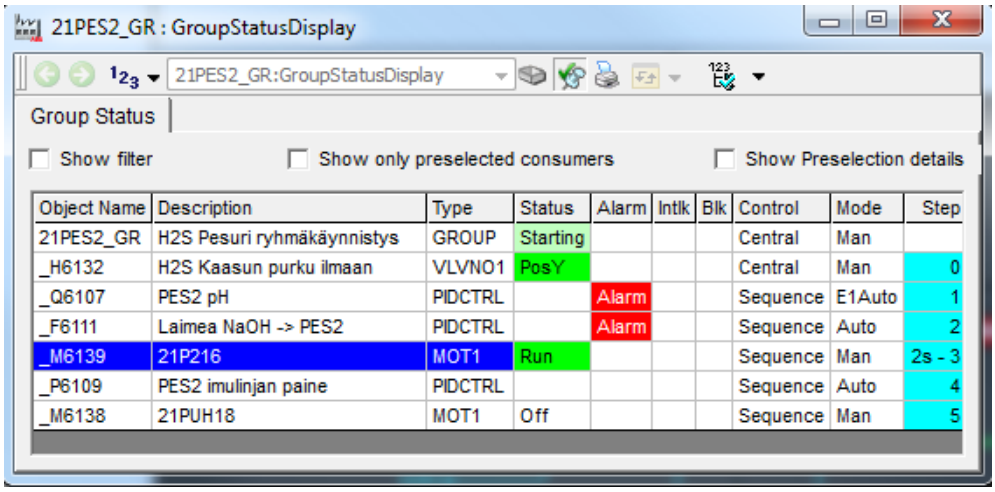
$Setp =$ setpoint value to the secondary controller
$Pmax =$ maximum parametrized range value of the secondary controller
$Pmin =$ minimum parametrized range value of the secondary controller
$Outp =$ Output value of the primary controller as percentage

4.2.4 System sequence start and stop

The system sequence start-and-stop function starts or stops the system motors and controllers in a set order with a single click from the operator screen. The function helps to ensure devices are started or stopped in the correct order, so as to prevent safety hazards, device breakdowns or environmental issues. The sequence was made with the built-in group function defined in the standard library of the ABB 800xA DCS.

The start and stop order is defined in the device-specific control modules (see Graph 19. on page 32). In the control module, there are five parameters used to define the device operation in a group sequence. First, there was defined to which group the device belongs to. A new data type GrpDrvCon was created under the desulphurization system data types. After this by defining the device control module with the parameter `IO.H2S_Kaasupesuri.GrpDrvCon` assigned the device to that group. Next the start and stop order of the devices were defined, for example P6109 was defined to be number 3, as seen on row 13 in Graph 19. The final parameters is the start and stop delays, used for example to assure the device is properly operating before starting the next device.

In the case of the scrubber, it was stated that the first device to be started was the cascade PID controller tagged Q6107 and F6111, as shown in Graph 24, in order to start adjusting the correct pH value of the cleansing solution. Next would be the pump M6139 so that the showers would start spraying the gas about to be fed in to the scrubber. The start condition for the pump M6139 was that the fluid level in the scrubber tank should be above 10%, as to the interlock discussed in section 4.2.2. If the interlock was active, the group start button would display that the device ‘M6139 is not ready for start’. After the scrubber would be successfully operating, the PID controller P6109 was initiated so the correct vacuum pressure in the gas flowline would be achieved. Finally the pump M6138 would start to vacuum the gas to the scrubber.


21PES2_GR : GroupStatusDisplay
21PES2_GR:GroupStatusDisplay
Group Status
Show filter
Show only preselected consumers
Show Preselection details
Object Name | Description | Type | Status | Alarm | Intlk | Blk | Control | Mode | Step
21PES2_GR | H2S Pesuri ryhmäkäynnistys | GROUP | Starting | | | | Central | Man |
_H6132 | H2S Kaasun purku ilmaan | VLVNO1 | PosY | | | | Central | Man | 0
_Q6107 | PES2 pH | PIDCTRL | | Alarm | | | Sequence | E1Auto | 1
_F6111 | Laimea NaOH -> PES2 | PIDCTRL | | Alarm | | | Sequence | Auto | 2
_M6139 | 21P216 | MOT1 | Run | | | | Sequence | Man | 2s - 3
_P6109 | PES2 imulinjan paine | PIDCTRL | | | | | Sequence | Auto | 4
_M6138 | 21PUH18 | MOT1 | Off | | | | Sequence | Man | 5

GRAPH 24. Group sequence status display

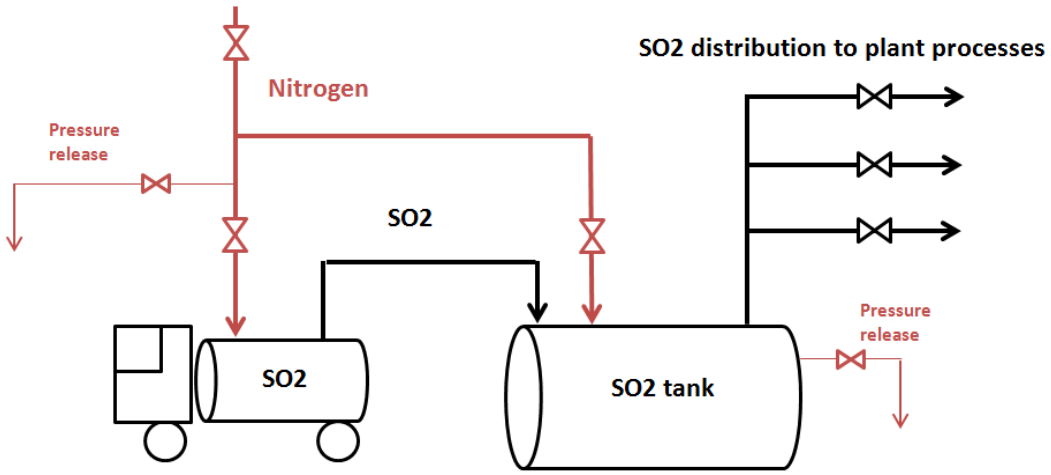
4.3 Summary

This chapter covered, in addition to operational insight of the H_2S desulphurization system, the sequence and methods with which a working automation application is created in the ABB's 800xA DCS. Also, in some extent, in other DCS systems as well. The practical programming phases-section also discussed the need for interlocks and control loops and related theory. Some of the program codes, for example interlock codes and cascade control codes, are not specific to the H_2S scrubber and can be utilized in other applications as well.

5 AUTOMATION OF SULFUR DIOXIDE DISTRIBUTION TANK

The second practical work in this thesis was to design, develop and program the automation of a sulfur dioxide distribution tank. Sulfur dioxide is a colorless, pungent smelling irritating gas that is in pressurized liquid form in the storage tank. At Freeport Cobalt, sulfur dioxide is used in processes to enhance dissolution of raw materials. Usage of the substance is mainly continuous and is quantitatively approximately 0,2 – 0,7 t/h, depending on the raw material situation. As shown in Graph 25, dry nitrogen gas is used to pressurize the SO_2 tank for transfer via pipelines to where needed. (Sirén Jarmo 2013.)

From the automation perspective, many safety features were needed because of the extreme toxicity of sulfur dioxide. Sulfur dioxide causes irritation of eyes, moist skin areas and respiratory tracts at 20ppm, coughing at 50-100ppm and in high concentrations, more than 400ppm, respiratory distress and risk of death. Long time exposure can aggravate susceptibility to respiratory diseases. (Sirén Jarmo 2013.)



GRAPH 25. Simplified operational description of the SO_2 distribution tank

5.1 Operation principle

The sulfur dioxide is brought to the site by a tanker truck. Two main hoses are then attached to the tanker, one for hydrogen pressurization of the truck container, and another for transfer into the main sulfur dioxide distribution storage tank. The hydrogen pressure in

the storage tank must be lowered in order to create a higher hydrogen pressure in the truck tank. An operator pushes a button in the operator environment screen of the SO_2 tank that automatically lowers the hydrogen pressure value in the tank. This forces the liquid sulfur dioxide from the truck into the storage tank. Control of the hydrogen pressure in the main storage tank is controlled by two PID loops, one for hydrogen feeding and another for outgoing SO_2 from the tank. These PID controllers are tagged P7312 and P7306 found in Appendix 2. Once the storage tank is filled, normal operating pressure is set back by an operator.

Other devices used are mainly for alarm and operation monitoring purposes. Several emergency valves are used to cut the flow of hydrogen into the tank and let out hydrogen pressure from the flowlines. Alarms for hazardous SO_2 content in the air are also used, these measurements show the content as parts-per-million.

5.1.1 Instrumentation

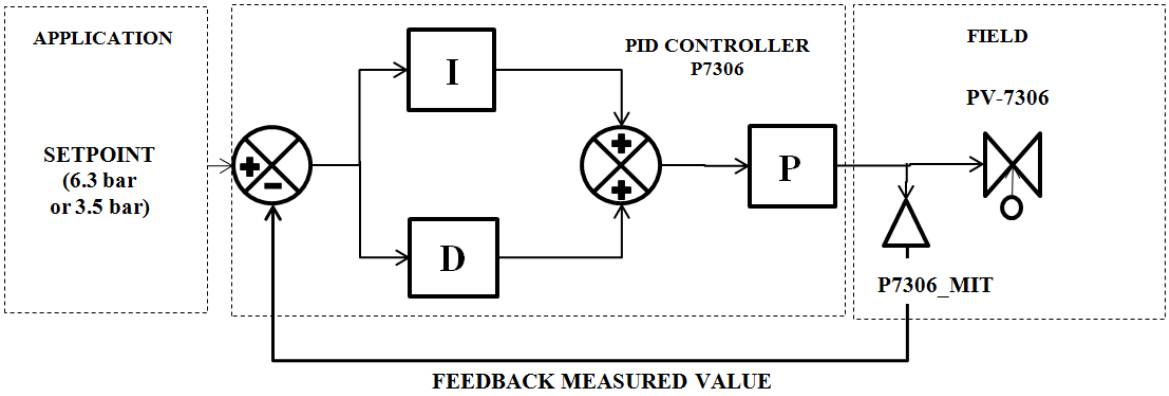
The entire operation of the SO_2 distribution process is based on using hydrogen pressure as the agent moving the sulfur dioxide. Therefore no electric motors are needed. The hydrogen pressure in the pipeline is at a constant value of 12.6bar when it reaches the SO_2 tank. The pressure is then regulated by control, and hand operated valves. As seen in Table 3. there are mostly pressure measurements for the purpose of alarms and interlocks.

TABLE 3. Instrument list of the SO_2 distribution tank

Tag	Description	AI	AO	DI	DO	Range	Unit
F7313	Liquid SO2 transfer nitrogen flow	1				0 - 20	bar
H7302	KULV emptying			1	1	on/off	
H7308	Liquid SO2 station emergency close			3		on/off	
L7305	Liquid SO2 tank level	1				0.3 - 3	m
P7301	Liquid SO2 line pressure 1	1				0 - 10	bar
P7302	SO2 pressure after break-film			1		on/off	3.5 bar
P7303	Liquid SO2 tank pressure 1					0 - 25	bar
P7304	Liquid SO2 transfer nitrogen pressure 1					0 - 10	bar
P7305	Differential pressure liquid SO2 tank			2	1	0 - 10	bar
P7306	Liquid SO2 tank pressure 2	1	1			0 - 15	bar
P7307	Liquid SO2 line pressure 2					0 - 10	bar
P7311	KU2 gas space pressure	1	1			-30..+10	mbar
P7312	Liquid SO2 tank nitrogen pressure		1			0 - 15	bar
P7313	SO2 transfer nitrogen pressure 2	1				0 - 20	bar
Q7306	Liquid SO2 tank-area gas alarm 1			1		0 - 15	ppm
Q7307	Liquid SO2 tank-area gas alarm 2			1		0 - 15	ppm
Q7312	Liquid SO2 tank-area gas alarm 3	1				0 - 25	ppm
Q7313	Liquid SO2 tank-area gas alarm 4	1				0 - 25	ppm
T7302	Liquid SO2 line temperature	1				-50..+100	°C
T7303	Liquid SO2 tank temperature 1					-50..+100	°C
T7306	Liquid SO2 tank temperature 2	1				-50..+100	°C
W7302	Liquid SO2 tank leveling	1				0 - 80	t

5.1.2 Control loops

There are two control loops, P7312 and P7306 utilized in control of the hydrogen pressure. Pressure is measured with sensors attached to the pipelines. The sensors give a feedback signal to the PID controllers which then control valves that constrict or open the flow accordingly. Graph 26, shows the logical structure of the PID controller P7306. The controller tagged P7312 is very similar to this. The setpoints of both controllers have two main values, this is for the purpose of creating a pressure difference when the tank is being refilled with SO_2. More of this in Section 5.2.3.



GRAPH 26. SO_2 distribution tank control loops

5.2 Practical programming phases

The program for the SO_2 was done under the application App_2 created previously (see Section 4.2). A new sub-application labeled SO2_Säiliö was created for the control modules and code specific to the SO_2 tank. The first step was to look over all the circuit diagrams for the devices to be programmed (see Appendix 5.). The control modules were then created and parameterized, in the same way as in Graph 19. Some conflicts between the circuit diagrams and the actual old programs were found. For example, there were devices documented that had not been in use for a long time. These were presumably caused by a lack of communication as the process had been changed over the years. However, these conflicts were quickly resolved.

Once the control modules had been programmed, the hardware definitions were added. This was done also in the same way as in Section 4.2. The I/O connection setups were received from the documentation designers and then added to the hardware structure in the Control Builder. All the instruments from Table 3. on page 42 were now defined to the DCS.

5.2.1 Process graphics display

The creation of the Human-Machine-Interface for the SO_2 tank was started by adding a new aspect in the Engineering Workplace. This graphics screen was created to correspond

to the PI diagram seen in Appendix 2. The screen shows the tank as the central part of the system and flowlines are represented as different colored lines connected to and from it. This should make it easier to get to know the system quickly without prior training of it. The dynamic objects, instrument measurements, valves and control loops were finally added to their corresponding positions for easy interaction and monitoring by the operators.

System interlock and operation description documents were written and created as button-links for quick interpretation of interlock events and operation overview. The created graphics screen can be seen in Appendix 4. The interlocks which were needed for this system can be found in the next section. The operational document explains the programming principles, i.e what happens when the "Täyttö" button on the screen is pressed. These two documents can be found in Appendices 8. and 9.

5.2.2 Interlocks

For this system two interlocks were needed for the filling operation of the tank. To prevent overfill of the SO_2 tank, an interlock cutting the nitrogen feed to the truck was programmed. The interlock had to have two conditions: If the pressure deviation between the tank and the incoming nitrogen is too low (below 1 bar) or if the tank level was at high limit of 2.6m or the weight of the tank at a high limit of 60t. When either of those conditions would be met, the interlock was to be put into effect. This would cause the valve P7305.1 to close (feeding of nitrogen) and the valve P7305.2 to open:

```
IO.SO2_Säiliö.P7305.In.IB1 := not IO.AIS.W7302_MIT.H2.Value;
IO.SO2_Säiliö.P7305.In.IB1 := not IO.AIS.L7305_MIT.H2.Value;

IF (IO.AIS.P7313_MIT.Signal.Value - IO.PID.P7306.MIT.Value) <= 1 then;
   IO.SO2_Säiliö.P7305.In.IB2 := false;
else;
   IO.SO2_Säiliö.P7305.In.IB2 := true;
end_if;
```

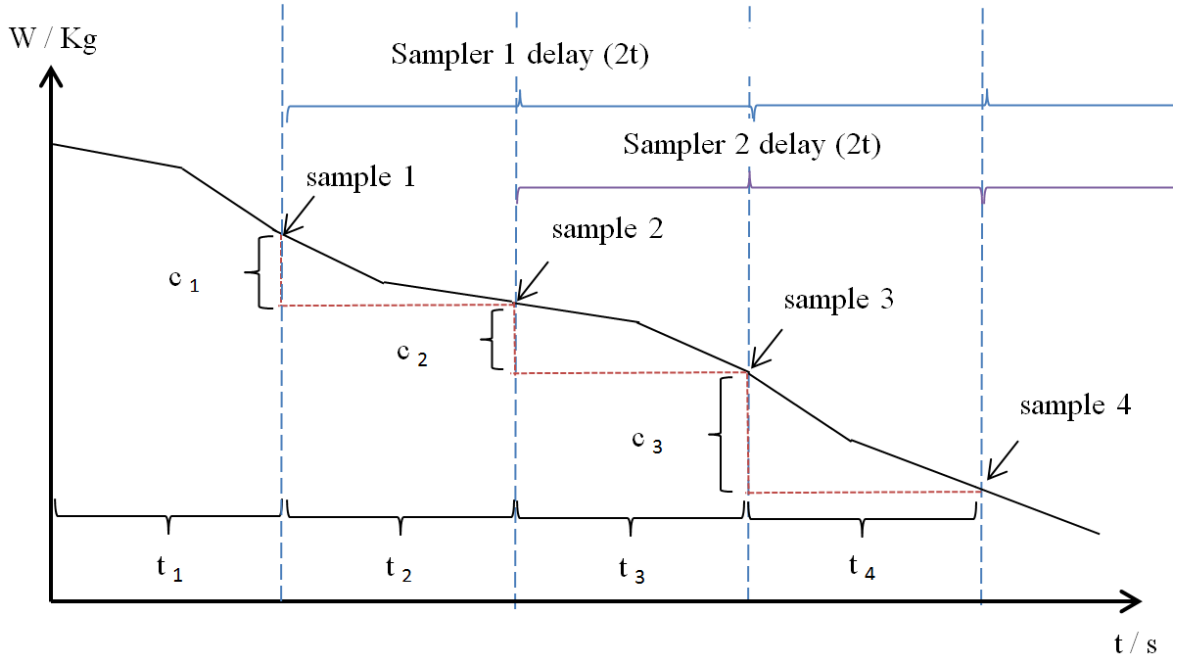
Additionally to the DCS interlock, there are also emergency stop buttons in the field which close and open valves by direct electric connections. When a stop button is pushed, the valve HV7308.1 closes the feed of nitrogen to both the tank and the truck, depressurizes the pipelines by opening the valve HV7308.2 and closes the feed of liquid SO_2 to the plant by opening HV7308.3. These valves have limit switches indicating they have been closed or opened. The limit switches are connected to the DCS by digital inputs and are displayed to the operator in the graphics screen for immediate information about this event. The circuit diagram of this can be seen in Appendix 10.

5.2.3 Calculated consumption measurement

An important fact to know about the SO_2 tank was the total consumption of the sulfur dioxide. This would show the amount of sulfur dioxide used by the plant processes at any given period of time. As discussed in the introduction of Section 5, the total consumption varies depending on the raw materials to be dissolved in the plant. This would also help to estimate the need for refills.

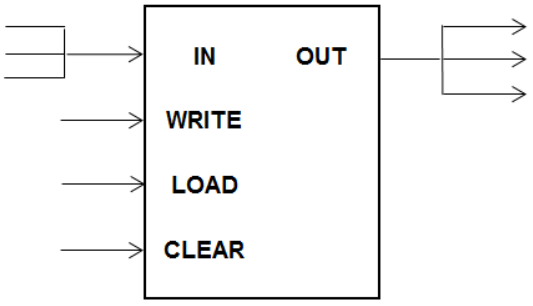
The boiling point for sulfur dioxide is -10°C and the melting point -72°C. Because the temperature in Finland might vary between -30°C in the winter and +30°C in the summer, the matter state of the sulfur dioxide is in liquid form in the winter and gas in the summer. From the automation point of view, this makes it hard to measure the flow of sulfur dioxide. Therefore no flow measurement was installed that could be used to show consumption.

The total consumption of the sulfur dioxide had to be measured with the decreasing weight of the tank over time. It was stated that the weight measurement of the tank was accurate and reliable and could thus be used to estimate the consumption. The programming principle was to take samples of the weight at fixed time intervals. Once two samples would be taken, the absolute differential result of these two values would show the consumption in the set time frame. Finally by multiplying the time by a necessary amount would give the consumption in kilograms per hour. This principle is illustrated in Graph 27.



GRAPH 27. Principle of the code used to calculate consumption of SO_2

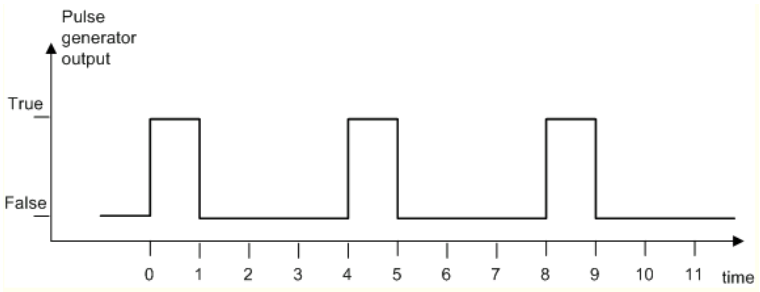
The process of designing the code was by trial and error of several different methods and Function Blocks. After several alternatives were developed, the most simple and elegant solution was selected. This solution was to introduce two Pulse Timer-Function Blocks, two Real Shift Register-Function Blocks and a Timer On-Function Block. The idea was that the signal from the weight measurement W7302_MIT would constantly be fed to the inputs of the two shift registers. In Graph 28, is a ShiftRegister Function Block. If the Load parameter is True, or if the Write parameter goes from False to True, the In[n] extensible inputs are copied to the Out[n] extensible outputs.



GRAPH 28. ShiftRegisterReal Function Block

The boolean signals to the Write parameter of the registers was fed in pulses of fixed time intervals by the two Pulse Timer Blocks (Graph 29.). Finally, the output of the shift

registers was differentiated and the absolute value of the result multiplied by 360 to get consumption value in kilograms per hour.



GRAPH 29. Timing diagram of PulseGenerator Function Block

As seen in Graph 27, the pulse signals do not overlap, because if they would, this would not show any deviation in the weight. To prevent the overlapping of the two sample intervals, a delay of one time-interval was introduced to the other sampler with the help of a Timer On-Function Block. The Timer Function Block gives an output Boolean signal after a set time delay has passed. The output of the timer is set as a condition for enabling the other shift register. This way the shift register only starts operation when the timer delay is finished. This delay is only initiated once when the controller starts operation. Below in Table 4. and the code following is the final Function Blocks and code used to achieve this measurement.

TABLE 4. Function Blocks used in the consumption measurement code

W7302_Pulssi	PulseGenerator	Sample1 interval
W7302_Pulssi2	PulseGenerator	Sample2 interval
W7302_Tall	RegisterReal[1]	Sample1 shifting to variable
W7302_Tall2	RegisterReal[1]	Sample2 shifting to variable
W7302_Viive	TOn	Sample2 enabling delay

```
if W7302_Viive.Q then
   W7302_Pulssi2_Enable := 1;
end_if;

W7302_Viive             (  In := true,
                           PT := SO2_Sailio_Times.D_10s);

W7302_Pulssi            ( PulseTime := SO2_Sailio_Times.D_1s,
                          Enable := true,
                          PeriodTime := SO2_Sailio_Times.D_20s );

W7302_Pulssi2           ( PulseTime := SO2_Sailio_Times.D_1s,
                          Enable := W7302_Pulssi2_Enable,
                          PeriodTime := SO2_Sailio_Times.D_20s );

W7302_Tall              ( In[1] := IO.AIS.W7302_MIT.Signal.Value,
                          Write := W7302_Pulssi.Out,
                          Out[1] => W7302_Nayte1);

W7302_Tall2             ( In[1] := IO.AIS.W7302_MIT.Signal.Value,
                          Write := W7302_Pulssi2.Out,
                          Out[1] => W7302_Nayte2);


W7302_Uusi :=   abs(W7302_Nayte1 - W7302_Nayte2);
IO.AIC.W7302_KULUTUS.Signal.IOValue := W7302_Uusi * 360.0;
```

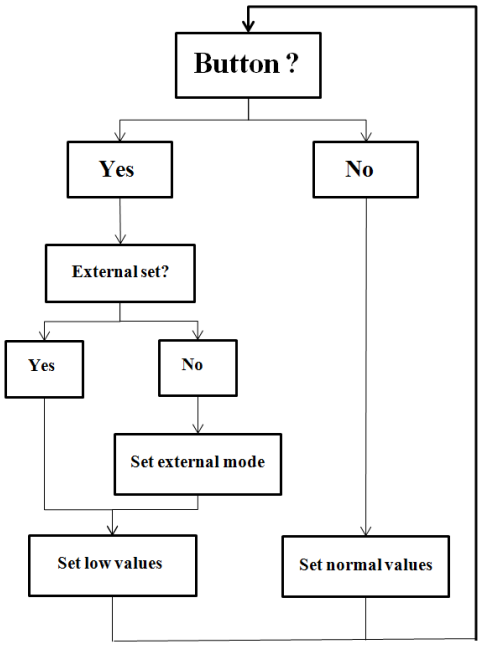
As seen from the final line of the code, the variable the final value is written to is of AIC data type (Analog Input Calculated). This is a separate Control Module that the system treats as if it were of normal AIS data type. The only difference is that the AIC signal comes from the code rather than an actual field instrument. The AIC measurement was also then imported into the process graphics. The final product can in theory be used for calculating the consumption of any tank or container with a level or weight measurement and constant consumption. Only thing needed to incorporate the code in to another application is to change the variables.

5.2.4 Automatic control loop setpoint drop

As described in section 5.1, the transfer of sulfur dioxide is done by applying constant nitrogen pressure in to the SO_2 tank. In tank filling situations, the same pressure is applied to the tanker truck SO_2. The pressure will initially be the same in both tanks and thus the SO_2 does not move. In order to move the SO_2, a pressure difference must be created between the truck and the SO_2 storage tank. The elements controlling the pressure in the

SO_2 tank are the control loops P7306 and P7312. The setpoint of these controllers must be dropped by -3bar for the refilling operation to be possible.

In a refilling operation, the operators were instructed to push a button labeled “Täyttö” (refill) in order to lower the SO_2 tank pressure. The button would lower the setpoints of the control loops P7306 and P7312. A program was then created for the automatic control loops setpoint drop. In order for the PID controllers to take external setpoint commands, external control mode would have to be set in both controllers. Once the background to this program was resolved, a flowchart seen in Graph 30, was created to clarify the event.



GRAPH 30. Flowchart describing the events in the automatic setpoint drop

The program for this was created in ST language using a conditional statement IF. As seen in the code below, the statement is initiated by the condition that the DOC (Digital Output Calculated) sends out a signal. This DOC, in the same way as in the consumption measurement, was created as a separate control module in the Control Builder and then placed as a button in the graphics screen.

```
if IO.DOC.SO2_TÄYTTÖ.Signal.Value then
   IO.SO2_Säiliö.P7306.In.SetExtAuto1 := 1;
   IO.SO2_Säiliö.P7312.In.SetExtAuto1 := 1;

   P7306_Arvo := 3.50;
   P7312_Arvo := 3.30;
else
   IO.SO2_Säiliö.P7306.In.SetExtAuto1 := 0;
   IO.SO2_Säiliö.P7312.In.SetExtAuto1 := 0;

   P7306_Arvo := 6.30;
   P7312_Arvo := 6.10;
end_if;
IO.SO2_Säiliö.P7306.In.ExtRefAuto1 := P7306_Arvo;
IO.SO2_Säiliö.P7312.In.ExtRefAuto1 := P7312_Arvo;
```

In the code, the variables P7306_Arvo and P7312_Arvo are used to store the setpoint values of the respective PID controllers. The first statement switches the control mode to external control in order for the values to be set as setpoints. The following else statement returns the values to normal and does no longer force external control once the button is turned off. The last lines of code explains that the external reference values are the variables P7306_Arvo and P7312_Arvo.

5.3 Summary

This chapter covered the automation case of the SO_2 distribution tank. It discussed the operational principle of the system and the programming related to it. The paragraph also gave more insight to how Structured Text applications are designed and finally programmed. In section 5.2.3 it becomes apparent how Function Blocks can be utilized in ST code in addition to traditional C language-like conditional statements.

6 CONCLUSIONS

In this thesis distributed control systems were studied and the automation programming of two process devices performed. The IEC 61131-3 programming languages were used for the programming of these devices in the ABB System 800xA DCS. The two devices, H_2S desulphurization system and SO_2 distribution tank were designed and programmed with user-friendliness, safety and operability as main priorities.

The first part of the programming work was to create a new application in the Control Builder structure and connect standard libraries to it. The libraries connected were the same used elsewhere in the plant, in other words, the BMI library components. In addition, there were some customized libraries created for use in Freeport Cobalt –specific cases which were also added. These customized libraries contained, for example, code for solution samplers used around the plant.

Next, the data types for each device were defined. Here, the first problems were encountered with the documentation of the devices connected to the AC2 controller to which all the instruments of the two device entireties were connected to. There were significant conflicts between the information found in the circuit and P&I diagrams compared to the existing process graphics and code of the two devices. These problems were resolved when ABB support personnel was contacted and asked to import the code from the old AC2 controller into a Microsoft Excel base. From this Excel file the actual used real-time parameters and devices could be found. The documentation was updated along the programming work. These conflicts were most likely caused by lack of communication between plant personnel: small changes to program code had been performed by several employees and subcontractors over the years and those changes were not forwarded to the circuit diagram designers. It was stated that up to date drawings would be very important for safety and operability. If a hazardous event would occur, explicit documentation of process devices would significantly speed up the maintenance work and accuracy.

Once the devices and their data types were successfully defined, programming work was started first from the H_2S desulphurization system. First, the operation of the system was studied and internalized. In the study, it was concluded that several safety and operability

features were lacking in the existing code. There was no documentation available in the process graphics display to explain interlocks or other system functions. Furthermore an interlock for the pump labeled M6139 was missing: it was possible to start the pump even without necessary fluid level in the tank. This would effectively cause the pump to break down. These problems were resolved by adding a group sequence start to the system that prevents unsafe starting conditions. The lacking documents were also created and made available for access directly from the process graphics screen.

The programming of the SO_2 distribution tank was started in the same way as the H_2S desulphurization system. Similar deficiencies were discovered: no documentation and inadequate interlocking and code. The process graphics display was also completely redesigned for better interpretation of the system. In the filling operation of the tank, the pressure had to be lowered in order to get a higher pressure in the filling truck as in the SO_2 tank. For this purpose, a button was programmed to ease this operation. The button automatically lowers the pressure instead of manually lowering it, thus ensuring correct operation values.

Operator friendliness is the key for effectiveness: no matter how good the automation programming is, it will always be the operator who is responsible for the operation of the system. The programmers will not constantly be around to supervise the operation, it is the task of the operators. This emphasizes the need for clear, simple process graphical displays or Human-Machine-Interfaces. The created applications were made as easy to use as possible: for example, the automatic control loop setpoint drop could have been done manually by separately setting the setpoints to the appropriate values. However, this would have left room for more human error as there are more things to take in to consideration. Now that the button is programmed, the only thing the operator needs to do is click one button that clearly states what it does: easy and efficient. Also the documents found in the appendices were solely made to aid the operators. If there is a confusing situation, they have quick and easy access to information related to the system. For example, from the interlock lists they can quickly see what will cause interlocks and how to prevent them from happening.

The PID controller loops were not thoroughly studied in this thesis. They have a very essential role in process automation and their effective operation is important for cost-

effectiveness. Classic control theory and its utilization at Freeport Cobalt were studied, but the PID parameters (gain, integration time and derivation time) were mostly predefined from the previous application and there was no need to tune the controllers. The retuning of the PID controllers might thus require further study at the plant.

The resulting control applications made the operation of both systems more efficient and safe. The operators would now have more tools to ease the operation of the systems and also have quick access to documentation regarding the interlocks and other matters related to operation procedures. The added interlocks will contribute to the safe operation of the systems if abnormal conditions occur in the future.

REFERENCES

ABB Inc. 2012a. System 800xA 5.1 Network Configuration. User manual.

ABB Inc. 2012b. System 800xA Engineering 5.1 Engineering and Production Environments. User manual.

ABB Inc. 2012c. System 800xA Engineering 5.1 Process Graphics. User manual.

ABB Inc. 2012d. System 800xA Control 5.1 AC 800M Planning. User manual.

ABB Inc. 2012e. System 800xA 5.1 System Guide Functional Description. User manual.

ABB Inc. 2003. released system document. Available: http://www05.abb.com/global/scot/scot296.nsf/veritydisplay/efa585f87b76f1a3c1256f6700017b0d/$file/3bus092080r0001_a_en_industrial_it_system_800xa_architecture_overview.pdf Accessed: 10. March 2013.

ABB Inc. 2013. System 800xA overview. Available: http://www.abb.com/product/seitp334/6d1455f13733373ac1257289002d2631.aspx Accessed: 14. March 2013.

Berge, J. 2002. Fieldbuses for Process Control: Engineering, Operation, and Maintenance. Durham: ISA

Davies, J. 2008. Windows Server 2008 TCP/IP Protocols and Services. Washington: Microsoft Press

Festo Didactic GmbH & Co. 1997. Process Control System workbook. Available: http://www.pacontrol.com/download/process-control-systems.pdf Accessed: 13. March 2013.

Fieldbus Foundation. 2006. description about Foundation technology. Available:

http://www.fieldbus.org/index.php?option=com_content&task=view&id=23&Itemid=308 Accessed: 7. March 2013.

Freeport-McMoran Copper & Gold Inc. 2013. Company profile. Available: http://www.fcx.com/company/who.html Accessed: 3. March 2013.

Funding Universe. 1997. OM Group Inc. history. Available: http://www.fundinguniverse.com/company-histories/om-group-inc-history/ Accessed: 3. March 2013.

King, M. 2011. Process Control – A practical approach. Chichester: Wiley

Korpijärvi, M. 2013. Interviews with Korpijärvi Marko, system programmer at Freeport Cobalt Ltd. between 1. March to 1. June 2013.

Syed, M. Soreanu, G. Falletta P. and Béland, M. 2006. Removal of hydrogen sulfide from gas streams using biological processes. Available: http://engrwww.usask.ca/oldsite/societies/csae/protectedpapers/c0436.pdf Accessed: 21. March 2013.

National Instruments Corp. 2012. Fundamentals, System Design, and Setup for the 4 to 20 mA Current Loop. Available: http://www.ni.com/white-paper/6940/en Accessed: 13. March 2013.

NITK Surathkal. 2011. Distributed Control System. Available: http://solve.nitk.ac.in/dmdocuments/electrical/DCS_write_up.pdf Accessed: 5. March 2013.

OMG Kokkola Chemicals Oy. 2011. Kestävää kemiaa. Available: http://www.kip.fi/omg/digipaper/index.html Accessed: 2. January 2013.

OPC Foundation. 2013. OPC description. Available: http://www.opcfoundation.org/Default.aspx/01_about/01_whatis.asp?MID=AboutOPC Accessed: 11. March 2013.

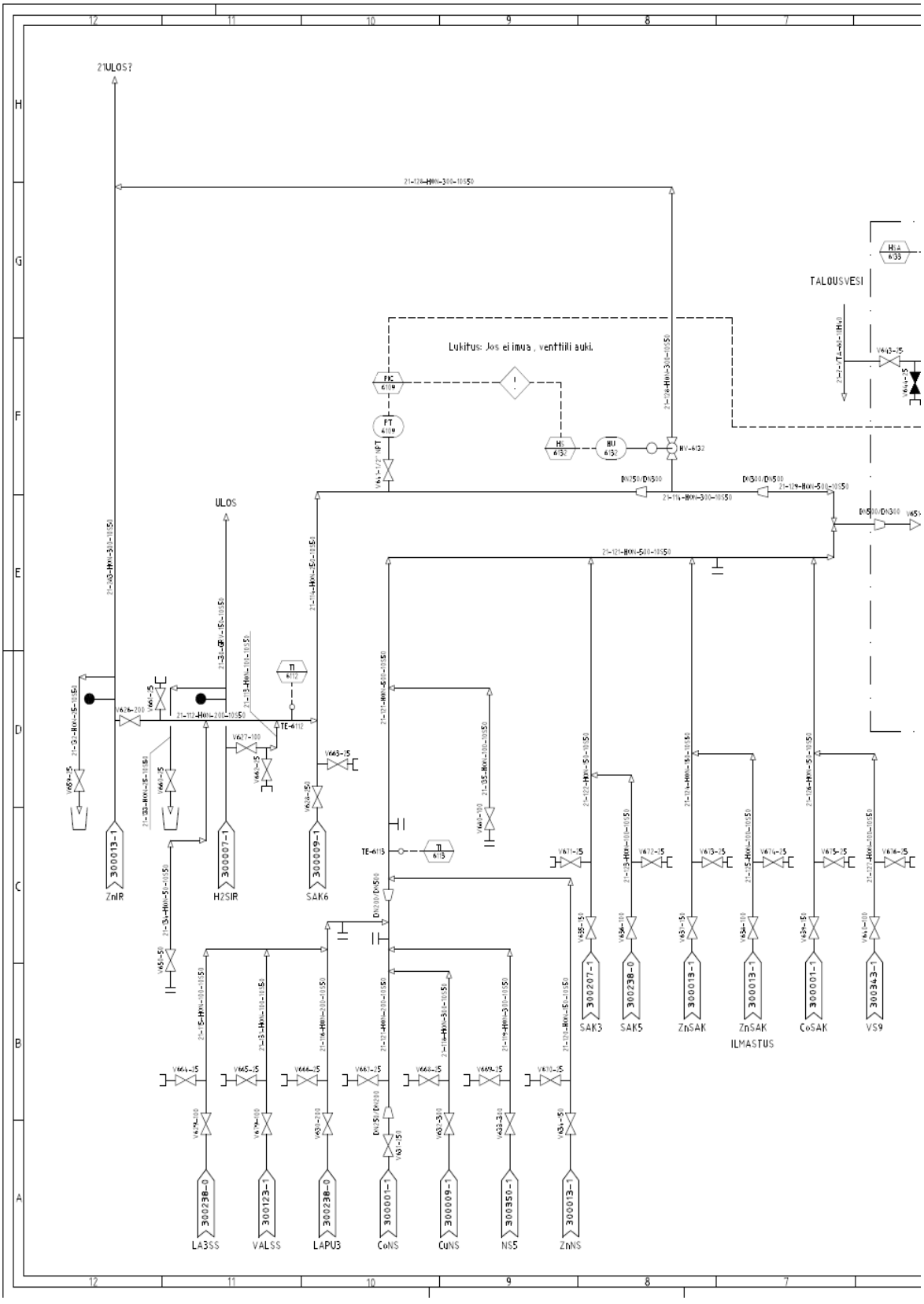
Paloranta, M. 2012. Automaatiojärjestelmä_EN_short.ppt. Presentation of automation system. Freeport Cobalt Ltd.

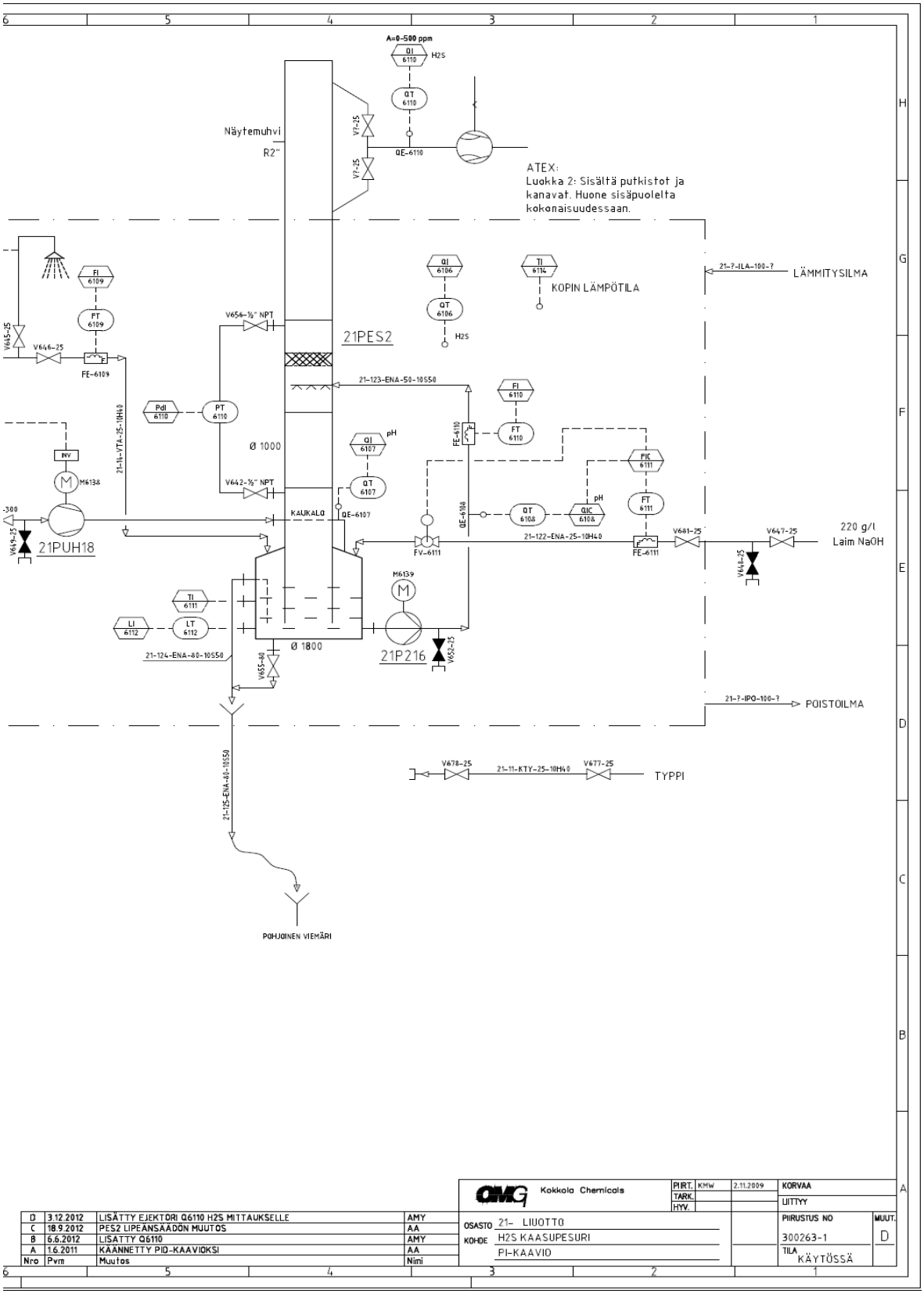
Paloranta, M 2013. Interviews with Paloranta Matias, process system engineer at Freeport Cobalt Ltd. between 1. March to 1. June 2013.

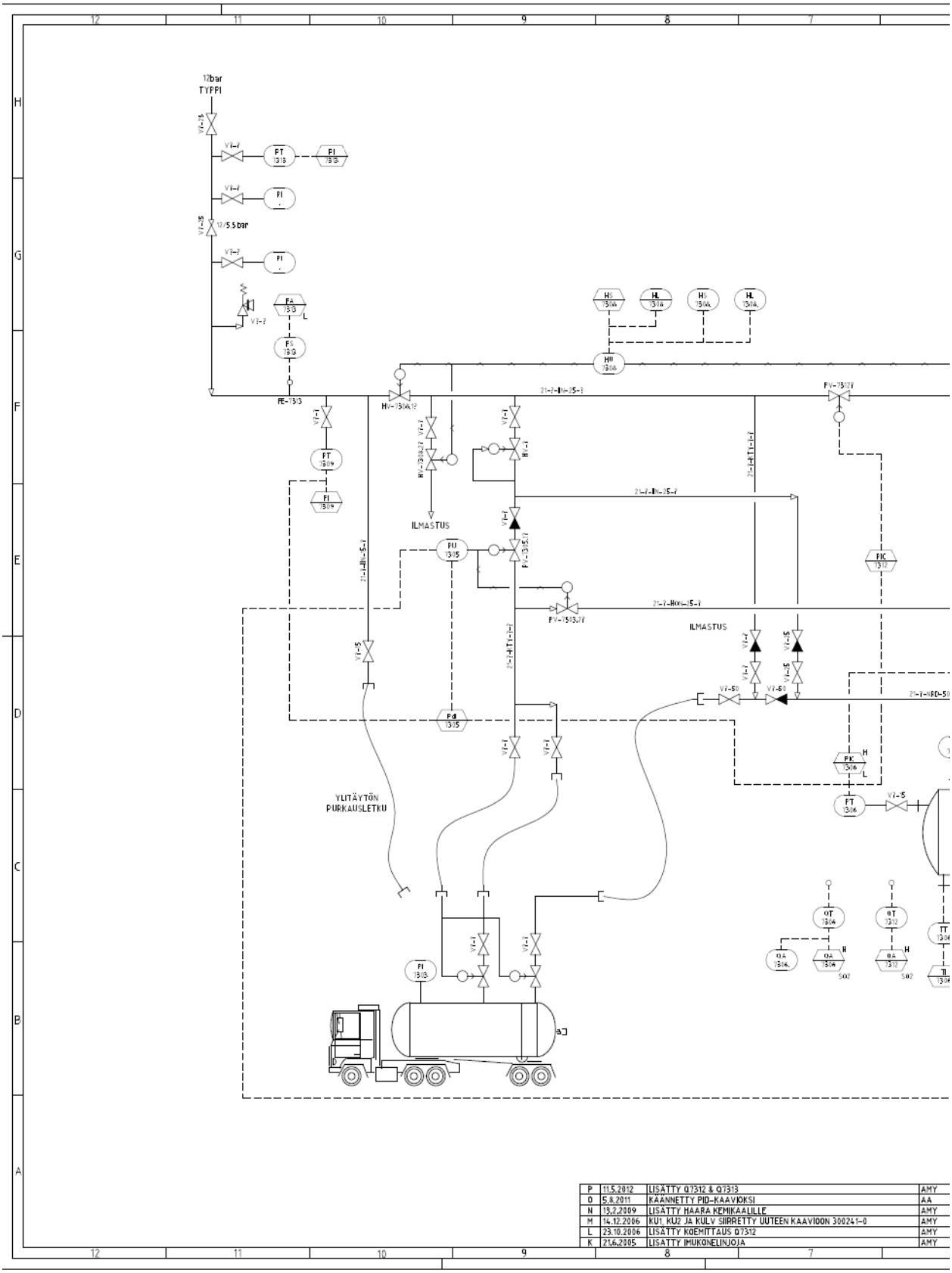
PI organization. 2013. PROFIBUS Technology and Application - System Description. Available: http://www.profibus.com/nc/downloads/downloads/profibus-technology-and-application-system-description/display/ Accessed: 7. March 2013.

Sirén, J. 2013. SO2 auton purkuohje OMG:lle. Work instruction manual.

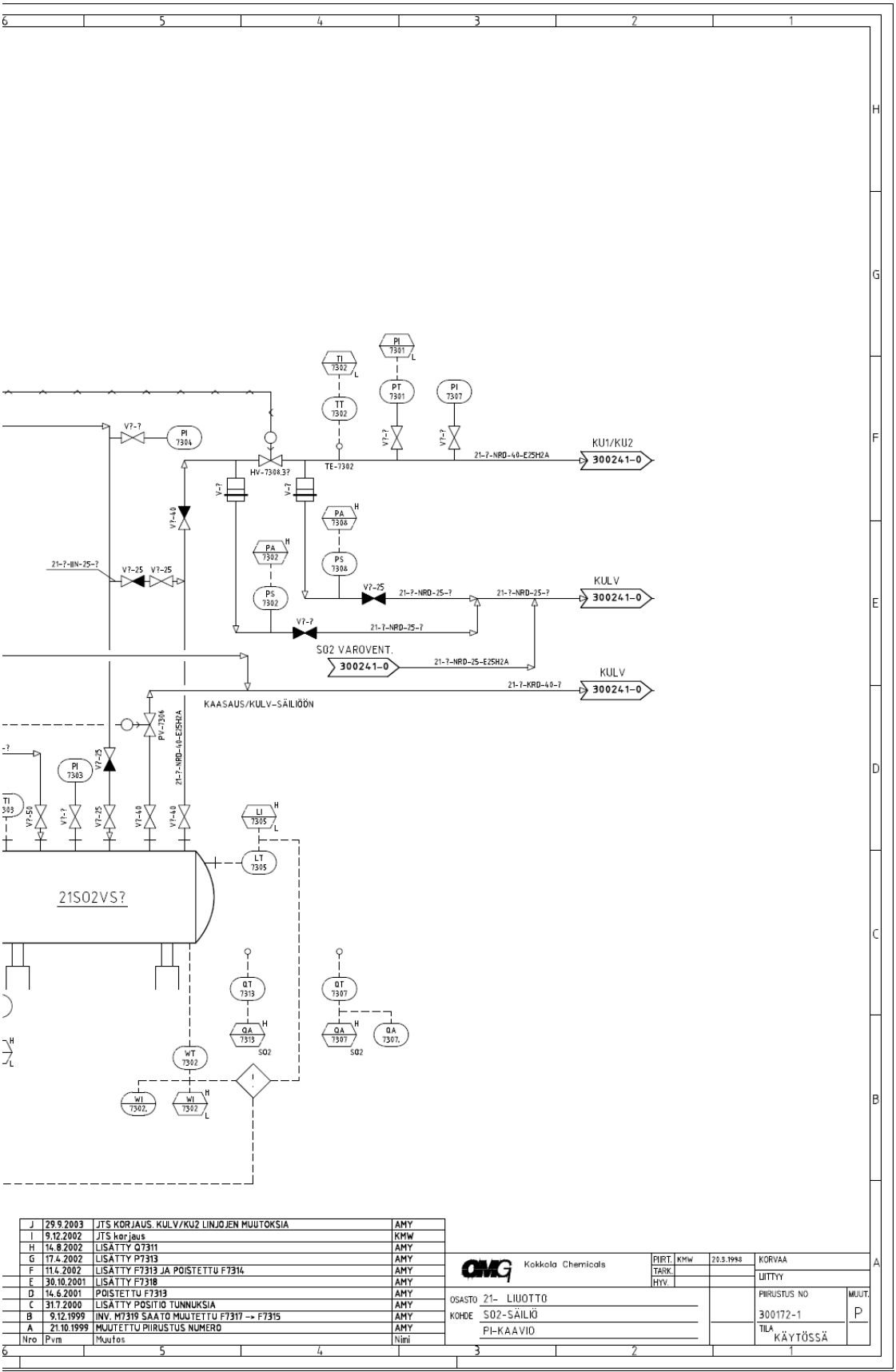
University of Alberta. 2003. Computer Organization and Architecture II. Available: http://webdocs.cs.ualberta.ca/~amaral/courses/329/webslides/Topic2-DeMorganLaws/sld017.htm Accessed: 14. March 2013


21ULOS?
TALOUSVESI
Lukitus: Jos ei ilmaa , venttiili auki.
ULOS
ZnIR
H2SIR
SAK6
SAK3
SAK5
ZnSAK
ZnSAK
ILMASTUS
CoSAK
VS9
LA3SS
VALSS
LAPU3
CoNS
CuNS
NS5
ZnNS


A=0-500 ppm
QI 6110
H2S
QT 6110
QE-6110
Näytemuhvi
R2"
V1-25
V1-25
ATEX:
Luokka 2: Sisältä putkistot ja kanavat. Huone sisäpuolelta kokonaisuudessaan.
FI 6109
FT 6109
FE-6109
V645-25
V646-25
V656-½" NPT
21PES2
QI 6106
QT 6106
H2S
TI 6114
KOPIN LÄMPÖTILA
21-?-ILA-100-?
LÄMMITYSILMA
21-123-ENA-50-10550
FI 6110
FT 6110
PdI 6110
PT 6110
Ø 1000
QI 6107
pH
QT 6107
V642-½" NPT
FE-6110
FIC 6111
QT 6108
QIC 6108
pH
FT 6111
KAUKALO
QE-6107
QE-6108
21-122-ENA-25-10H40
V681-25
V647-25
220 g/l
Laim NaOH
FE-6111
FV-6111
V646-25
INV
M6138
21-14-VTA-25-10H40
-300
V649-25
21PUH18
M6139
TI 6111
LI 6112
LT 6112
21-124-ENA-80-10550
Ø 1800
21P216
V652-25
V655-80
21-?-IPO-100-?
POISTOILMA
V678-25
21-11-KTY-25-10H40
V677-25
TYPPI
21-125-ENA-80-10550
POHJOINEN VIEMÄRI
Kokkola Chemicals
PIRT. KMW 2.11.2009
TARK.
HYV.
KORVAA
LIITTYY
OSASTO 21- LIUOTTO
KOHDE H2S KAASUPESURI
PI-KAAVIO
PIIRUSTUS NO 300263-1
MUUT. D
TILA KÄYTÖSSÄ
D 3.12.2012 LISÄTTY EJEKTORI Q6110 H2S MITTAUKSELLE AMY
C 18.9.2012 PES2 LIPEÄNSÄÄDÖN MUUTOS AA
B 6.6.2012 LISÄTTY Q6110 AMY
A 1.6.2011 KÄÄNNETTY PID-KAAVIOKSI AA
Nro Pvm Muutos Nimi



P	11.5.2012	LISÄTTY Q7312 & Q7313	AMY
O	5.8.2011	KÄÄNNETTY PID-KAAVIOKSI	AA
N	13.2.2009	LISÄTTY HAARA KEMIKAALILLE	AMY
M	14.12.2006	KU1, KU2 JA KULV SIIRRETTY UUTEEN KAAVIOON 300241-0	AMY
L	23.10.2006	LISÄTTY KOEMITTAUS Q7312	AMY
K	21.6.2005	LISÄTTY IMUKONELINJOJA	AMY



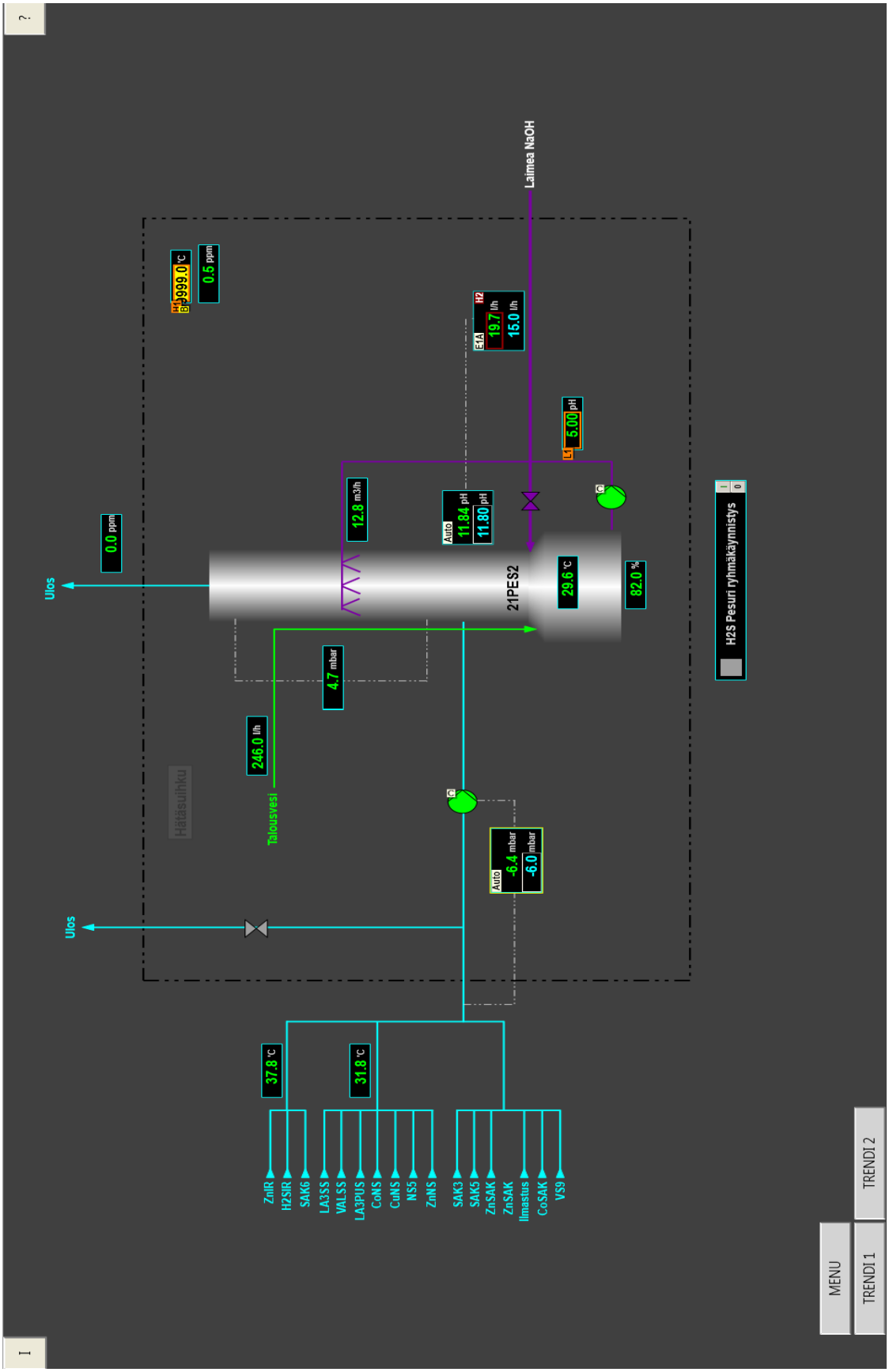
Nro	Pvm	Muutos	Nimi
J	29.9.2003	JTS KORJAUS. KULV/KU2 LINJOJEN MUUTOKSIA	AMY
I	9.12.2002	JTS korjaus	KMW
H	14.8.2002	LISÄTTY Q7311	AMY
G	17.4.2002	LISÄTTY P7313	AMY
F	11.4.2002	LISÄTTY F7313 JA POISTETTU F7314	AMY
E	30.10.2001	LISÄTTY F7318	AMY
D	14.6.2001	POISTETTU F7313	AMY
C	31.7.2000	LISÄTTY POSITIO TUNNUKSIA	AMY
B	9.12.1999	INV. M7319 SAATO MUUTETTU F7317 --> F7315	AMY
A	21.10.1999	MUUTETTU PIIRUSTUS NUMERO	AMY

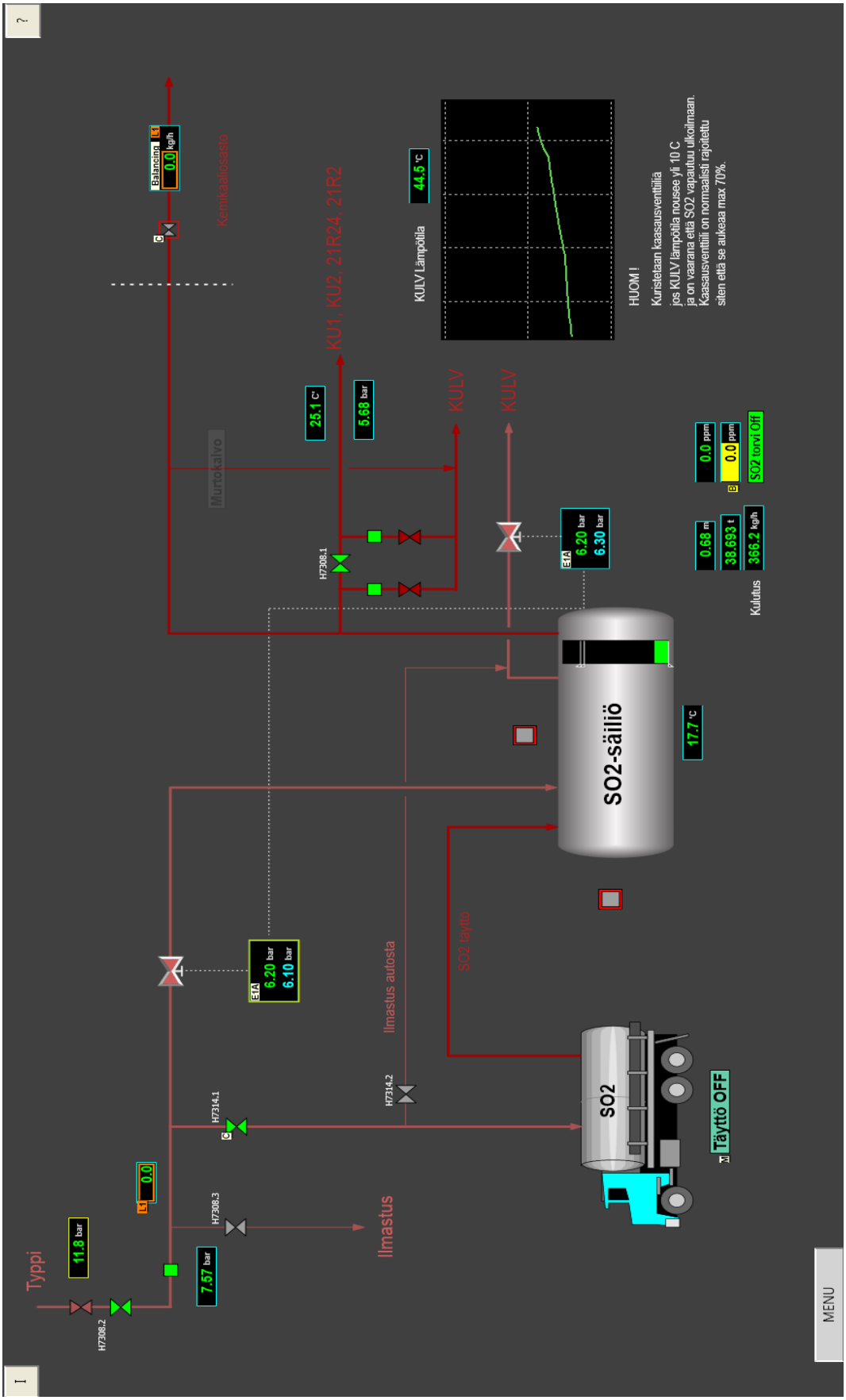
Kokkola Chemicals

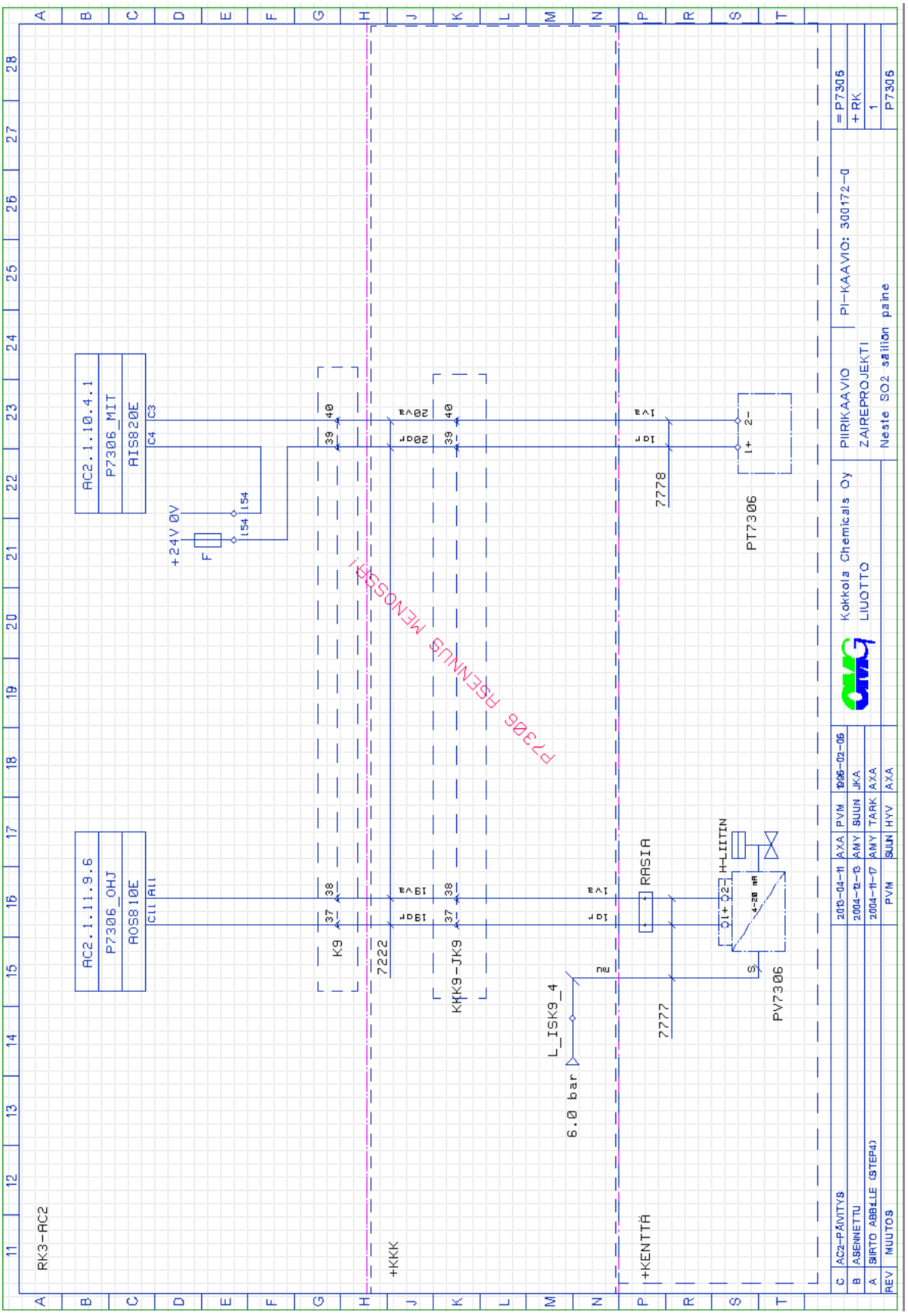
PIRT.	KMW	20.3.1998
TARK.		
HYV.		

OSASTO 21- LIUOTTO
KOHDE SO2-SÄILIÖ
PI-KAAVIO

KORVAA
LIITTYY
PIIRUSTUS NO 300172-1
MUUT. P
TILA KÄYTÖSSÄ


Laimea NaOH
0.5 ppm
999.0 °C
19.7 l/h
15.0 l/h
6.00 pH
12.8 m3/h
11.84 pH
11.80 pH
0.0 ppm
Ulos
21PES2
29.6 °C
82.0 %
H2S Pesuri ryhmäkäynnistys
4.7 mbar
246.0 l/h
Talousvesi
Hätäsuihku
-6.4 mbar
-6.0 mbar
Ulos
37.8 °C
31.8 °C
ZnIR
H2SIR
SAK6
LA3SS
VALSS
LA3PUS
CoNS
CuNS
NS5
ZnNS
SAK3
SAK5
ZnSAK
ZnSAK
Ilmastus
CoSAK
VS9
MENU
TRENDI 1
TRENDI 2


Balancing
0.0 kg/h
Kemikaaliosasto
KU1, KU2, 21R24, 21R2
44.5 °C
KULV Lämpötila
HUOM !
Kuristetaan kaasausventtiilia
jos KULV lämpötila nousee yli 10 C
ja on vaarana että SO2 vapautuu ulkoilmaan.
Kaasausventtiili on normaalisti rajoitettu
siten että se aukeaa max 70%.
Murtokalvo
25.1 c°
5.68 bar
KULV
KULV
H7308.1
EIA
6.20 bar
6.30 bar
0.0 ppm
0.0 ppm
SO2 torvi Off
0.68 m
38.693 t
366.2 kg/h
Kulutus
SO2-säiliö
17.7 °C
SO2 täyttö
Ilmastus autosta
H7314.2
H7314.1
SO2
Täyttö OFF
EIA
6.20 bar
6.10 bar
0.0
H7308.3
Ilmastus
11.8 bar
7.57 bar
Typpi
H7308.2
MENU


AC2.1.11.9.6
P7306_OHJ
AOS810E
AC2.1.10.4.1
P7306_MIT
AIS820E
+24V 0V
F
K9
7222
KKK9-JK9
7778
PT7306
RASIA
7777
L_ISK9_4
6.0 bar
H-LIITIN
PV7306
P7306 RSENNUS MENOSSÄ!
RK3-AC2
+KKK
+KENTTÄ
Kokkola Chemicals Oy
LUOTTO
PIIRIKAAVIO
ZAIREPROJEKTI
Neste SO2 sallion paine
PI-KAAVIO: 300172-0
= P7306
+ RK
P7306
AC2-PÄIVITYS
ASENNETTU
SIIRTO ABB&LLE (STEP4)
MUUTOS
PVM
1996-02-06
2003-04-11
2004-12-13
2004-11-17
JKA
AXA
AMY
TARK
SUUN
HYV

21PES2 Lukitukset

Kaasun purku ulkoilmaan mikäli imulinjassa ei tarpeeksi alipainetta:

(IB1 10 sekunnin lukitusviive)

Positio	Lukitus	Kuvaus
H6132	P6109_MIT > H2	H2S kaasun purku ilmaan

Pumpun M6139 käynnistyksen esto mikäli säiliössä ei tarpeeksi nestettä:

(IB1 10 sekunnin lukitusviive)

Positio	Lukitus	Kuvaus
M6139	L6112_MIT <L2	21P216, pesurin kierto

21PES2 ryhmäkäynnistys

Askel	Positio	Toiminto	Kuvaus
1.	Q6107	E1A-tilaan	21PES2 pH säätö
2.	F6111	AUTO-tilaan	Laimea NaOH -> PES2
3.	M6139	Käyntiin	21P216 kiertopumppu
4.	P6109	AUTO-tilaan	21PES2 imulinjan paineensäätö
5.	M6138	Käyntiin	21PUH18 imulinjan puhallin

Pysäytys päinvastaisessa järjestyksessä.

M6139 käynnistysehtona on ettei pinta L6112_MIT ole liian matala <L2. Säiliössä täytyy siis olla nestettä tarpeeksi ennen käynnistystä.

SO2-Säiliön lukitukset

Säiliön täyttöön liittyvät lukitukset:

Lukitukset jolloin auton purku keskeytyy: Typpi sulkeutuu autolle, P7305 kiinni ja P7305_RKI auki:

1. Jos paine-ero säiliön ja tulevan typpipaineen välillä on liian pieni (<1bar)
2. Jos säiliön pinta (H2 = 2,6m) tai paino (H2 = 60t) menee ylärajalukitukseen (ylitäytön esto)

Positio	Lukitus
P7305	W7302 tai L7305 > H2 tai P7313 ja P7306 välinen paine-ero <= 1bar

SO2 hätäseispainikkeiden lukitus:

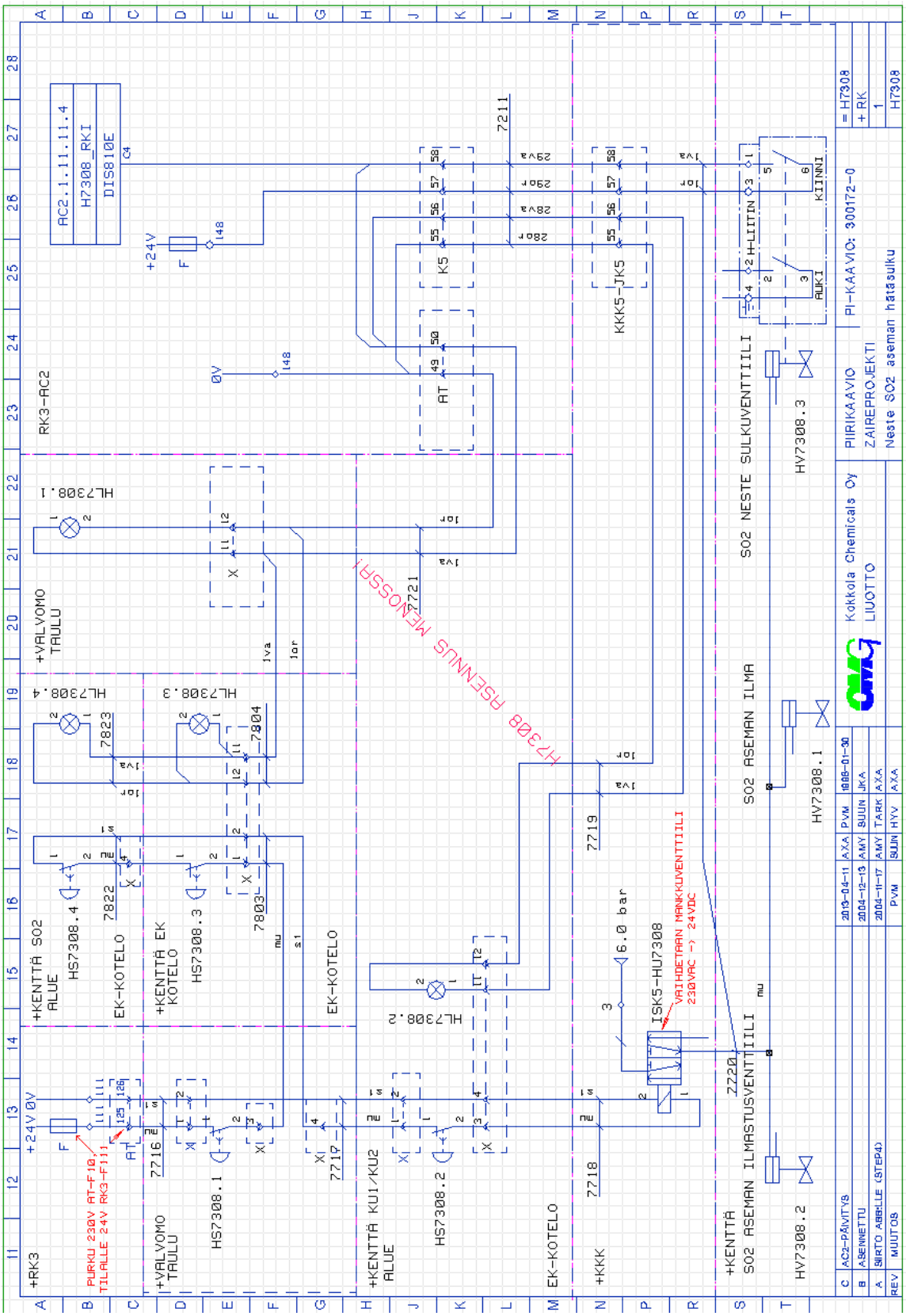
Jos joku hätäseispainikkeista aktivoituu, suljetaan typpi SO2-säiliölle ja autolle (H7308_RKI2), päästetään paineet pois typpilinjasta (H7308_RKI3) ja suljetaan nestemäisen SO2:n syöttäminen prosessiin (H7308_RKI)

Täyttö-painikkeen toiminta

Kun täyttöpainike painetaan "Täyttö ON" tilaan, pudottaa tämä paineen säädön ja kaasauksen säätöpiirit (P7312 ja P7306) alemmas arvoihin 3,3 bar ja 3,5 bar. Painikkeen aktivointi asettaa myös säätöpiirit E1A-tilaan, mikäli säätöpiirit on automaatilla (Auto), tai missä tahansa muussa tilassa. Painikkeen asettaminen takaisin "Täyttö OFF" tilaan ei kuitenkaan aseta takaisin säätöpiirien edellistä toimintatilaa, vaan se asetetaan käsin. "Täyttö OFF" tilassa säätöpiirien P7312 ja P7306 asetusarvot asettuvat arvoihin 6,1 bar ja 6,3 bar.

SO2 torvi

Säiliöalueen kaasuhälytys (Q7312_TORVI) aktivoituu kun Q7312_MIT saavuttaa H2 rajan 8.0 ppm. Torvin saa myös käsitilaan.


AC2.1.11.11.4
H7308_RKI
DIS810E
RK3-AC2
+24V
0V
HL7308.1
HL7308.2
HL7308.3
HL7308.4
+VALVOMO TAULU
+KENTTÄ SO2 ALUE
HS7308.4
EK-KOTELO
+KENTTÄ EK KOTELO
HS7308.3
HS7308.1
HS7308.2
+KENTTÄ KU1/KU2 ALUE
+KKK
+KENTTÄ SO2 ASEMAN ILMASTUSVENTTIILI
HV7308.1
HV7308.2
HV7308.3
SO2 ASEMAN ILMA
SO2 NESTE SULKUVENTTIILI
H7308 PÄÄNSULKU
KK5-HU7308
VAIHDETAAN MANKKUVENTTIILI 230VAC -> 24VDC
6.0 bar
KKK5-JK5
K5
AT
H-LIITIN
AUKI
KIINNI
PURKU 230V AT-F10
TILALLE 24V RK3-F11
Kokkola Chemicals Oy
PIIRIKAAVIO
ZAIREPROJEKTI
Neste SO2 aseman hätäsulku
PI-KAAVIO: 300172-0
= H7308
+ RK
1
H7308
LIUOTTO
AC2-PÄIVITYS
ASENNETTU
SIIRTO ABBELLE (STEP4)
MUUTOS
REV
PVM
2013-04-11
2004-12-13
2004-11-17
1998-01-30
AXA
AMY
TARK
HYV

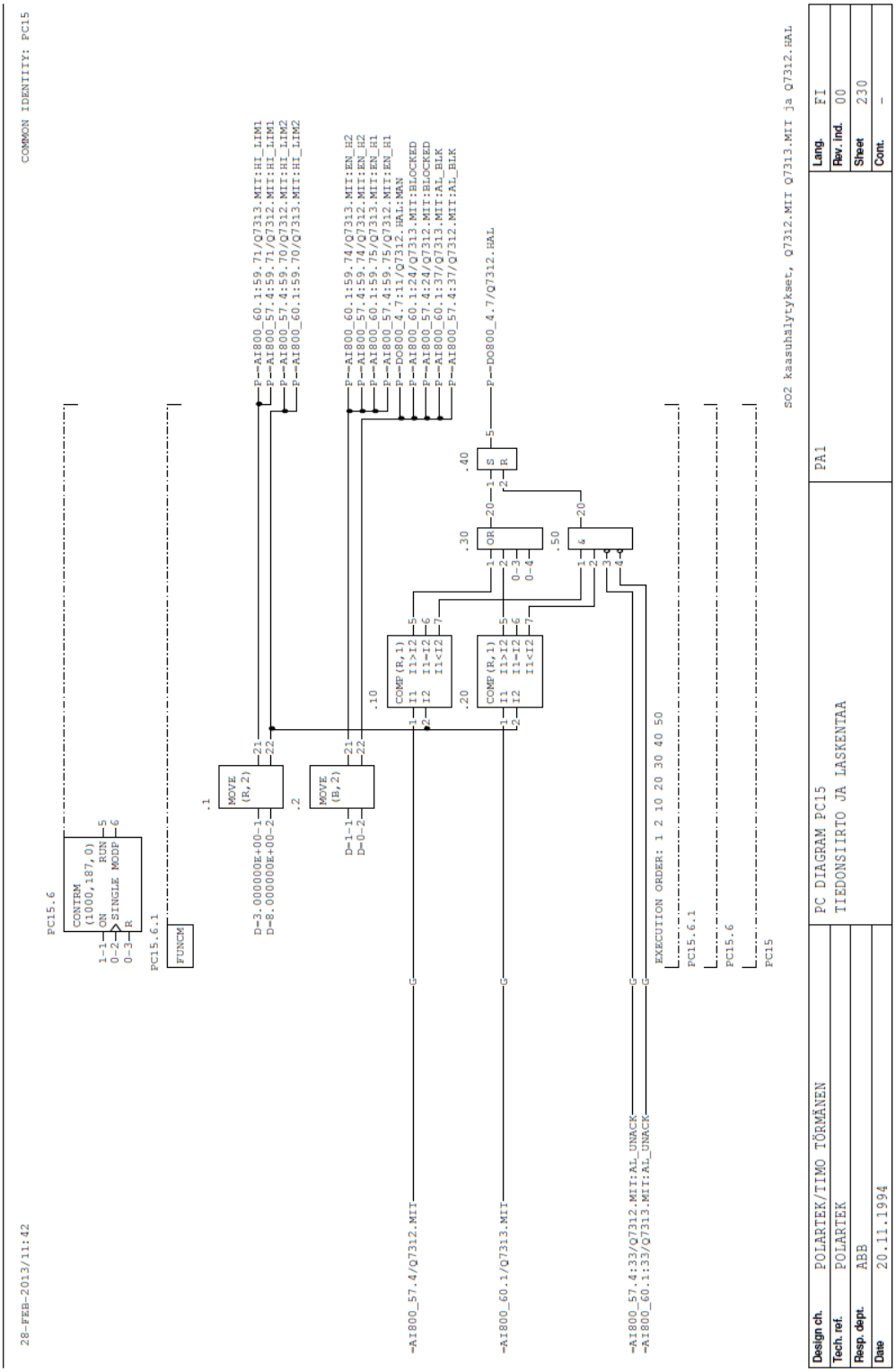


ABB Automation
28-FEB-2013/11:42
COMMON IDENTITY: PC15
PC15.6
CONTRM
(1000,187,0)
ON
RUN
SINGLE MODP
R
PC15.6.1
FUNCM
.1
MOVE
(R,2)
D=3.000000E+00-1
D=8.000000E+00-2
.2
MOVE
(B,2)
D-1-1
D-0-2
.10
COMP(R,1)
I1 I1>I2
I2 I1=I2
I1<I2
.20
COMP(R,1)
I1 I1>I2
I2 I1=I2
I1<I2
.30
OR
.40
S
R
.50
&
P--AI800_60.1:59.71/Q7313.MIT:HI_LIM1
P--AI800_57.4:59.71/Q7312.MIT:HI_LIM1
P--AI800_57.4:59.70/Q7312.MIT:HI_LIM2
P--AI800_60.1:59.70/Q7313.MIT:HI_LIM2
P--AI800_60.1:59.74/Q7313.MIT:EN_H2
P--AI800_57.4:59.74/Q7312.MIT:EN_H2
P--AI800_60.1:59.75/Q7313.MIT:EN_H1
P--AI800_57.4:59.75/Q7312.MIT:EN_H1
P--DO800_4.7:11/Q7312.HAL:MAN
P--AI800_60.1:24/Q7313.MIT:BLOCKED
P--AI800_57.4:24/Q7312.MIT:BLOCKED
P--AI800_60.1:37/Q7313.MIT:AL_BLK
P--AI800_57.4:37/Q7312.MIT:AL_BLK
P--DO800_4.7/Q7312.HAL
-AI800_57.4/Q7312.MIT
-AI800_60.1/Q7313.MIT
-AI800_57.4:33/Q7312.MIT:AL_UNACK
-AI800_60.1:33/Q7313.MIT:AL_UNACK
EXECUTION ORDER: 1 2 10 20 30 40 50
PC15.6.1
PC15.6
PC15
SO2 kaasuhälytykset, Q7312.MIT Q7313.MIT ja Q7312.HAL
Design ch. POJARTEK/TIMO TÖRMÄNEN
Tech. ref. POJARTEK
Resp. dept. ABB
Date 20.11.1994
PC DIAGRAM PC15
TIEDONSIIRTO JA LASKENTAA
PA1
Lang. FI
Rev. ind. 00
Sheet 230
Cont. -